FM TOWNS

FUJITSU FM SERIES PERSONAL COMPUTER

386 | DOS-Extender™
ユーザーズマニュアル



FUJITSU

FUJITSU

81SP-0072-1

FM TOWNS 386 | DOS-Extender™ユーザーズマニュアル 81SP-0072-1

FM TOWNS

386 | DOS-Extender™
ユーザーズマニュアル

富士通株式会社

ごあいさつ

このたびは「FM TOWNS 386 | ASM TOOL KIT V1.1」をお買い上げいただき、まことにありがとうございます。

386 | DOS-Extender™ は、MS-DOS のもとで実行するアプリケーションプログラムに80386のネイティブモードの実行環境を提供しています。本書では、この 386 | DOS-Extender™ のもとで 80386 ネイティブモードのプログラムを実行する方法について説明し、さらに 80386 ネイティブモード環境についても詳細に説明しています。

本マニュアルには、“外国為替及び外国貿易管理法”に基づく特定技術が含まれています。したがって、本マニュアルまたはその一部を輸出する場合には、同法に基づく許可が必要とされます。

1989 年 10 月

■開発キットのマニュアル構成と本書の位置づけ

FMTOWNS 上で、C言語やアセンブラでのソフトウェア開発を可能にするため、富士通では「FMTOWNS 386 | ASM TOOL KIT V1.1」と「FMTOWNS High C™ Compiler V1.4」の２つの関連した開発キットを用意しております。

以下に示すのはこれら２つの開発キットに添付されているマニュアルです。各マニュアルの関連性を考え、本書をお読みください。

製品名称：　FMTOWNS 386 | ASM TOOL KIT V1.1

マニュアル	内容
アプリケーション開発キット ユーザーズガイド	開発キットで提供されるソフトウェアやマニュアルの概要、FMTOWNS でアプリケーションを開発する環境や手順についての概要など全体を見渡せる情報が記載されています。
386 | DOS-Extender™ ユーザーズマニュアル	386 | DOS-Extender™のもとで 80386ネィティブモードのプログラムを実行する方法について説明し、さらに 80386ネイティブモード環境についても詳細に解説しています。
386 | ASM TOOL KIT ユーザーズマニュアル	FMTOWNS 上でアセンブラを使用してアプリケーションプログラムを開発するユーザのために、開発の過程で必要な各ソフトウェアの使用方法を詳細に解説しています。 次の４つの部で構成されています。 第１部　アセンブラ　386 | ASM 第２部　リンカ　386 | LINK 第３部　シンボリックデバッガ　386 | DEBUG 第４部　ライブラリアン　386 | LIB

注）網掛けで示されたマニュアルが本書です。

TownsBIOS
リファレンス

FMTOWNS の持つBIOS機能について、リファレンス形式で詳細に解説しています。
BIOSの機能により、次の６つの章に分類されています。

1 グラフィックBIOS　　4 フォントBIOS
2 マウスBIOS　　5 CD-ROM BIOS
3 スプライトBIOS　　6 サウンドBIOS

製品名称：　FMTOWNS　High C™ Compiler V1.4

High C™ Compiler
ユーザーズマニュアル

FMTOWNS でHigh C™ Compiler を使用してアプリケーションプログラムを開発するユーザのために、プログラム開発過程、High Cの文法、High Cの標準ライブラリについて解説しています。
次の３部で構成されています。

第１部　プログラマーズガイド
第２部　ランゲージリファレンス
第３部　標準ライブラリ

Cライブラリ
リファレンス

FMTOWNS 専用のCライブラリについて、その機能や、使用方法、使用例などをリファレンス形式で詳細に解説しています。
ライブラリ関数の機能により、次の８つの章に分類されています。

1 グラフィック　　5 CD-ROM
2 マウス　　6 サウンド
3 スプライト　　7 EUPHONY ドライバ
4 フォント　　8 キーボード

目次

FM TOWNS
386|DOS−Extender™
ユーザーズマニュアル

第1章　概要

386｜DOS-Extender（以降はDOS-Extenderとします）は、MS-DOSの上で実行されるアプリケーションプログラムに、 80386のネイティブモードの実行時環境を提供します。

386 ネイティブモードで実行するプログラムは、パソコンのすべてのメモリ（1MBを超えるメモリを含めて）を直接アクセスできます。

DOS-ExtenderはMS-DOSとアプリケーションプログラム間の中継層として機能します。アプリケーションプログラムはソフトウェア割込みを使った標準的な方法でMS-DOSとBIOSのシステムコールを行います。DOS-Extenderは、このシステムコールをインタセプトし、 386のリアルモードに切り換えMS-DOSに受け渡します。さらにDOS-Extenderは、リアルモードと 386ネイティブモード間でのデータのやりとりの処理を自動的に行うので、プログラムの大きさやメモリ構成に関係なく、MS-DOSの機能を利用することができます。

DOS-Extenderによって、MS-DOSは 386ネイティブモードのオペレーティングシステムとして機能するようになります。

本書では、DOS-Extenderを使い 386ネイティブモードでプログラムを実行する方法を説明します。

また、本書では、プログラムをDOS-Extenderの制御下で実行する 386ネイティブモード環境について専門的に解説します。本書の説明は、専門的かつ詳細になるので、読者はアプリケーションプログラムの作成に必要な情報だけを選んで読んでください。第1章から第5章には、ほとんどの 386ネイティブモードアプリケーションプログラムを作成するのに十分な情報が含まれています。本書では、MS-DOSのリアルモード実行環境とDOS-Extenderを加えたMS-DOSの 386ネイティブモード実行環境の違いについて詳しく説明しますが、本書の読者は8086のプログラミング方法とMS-DOSに精通していることを前提にして、MS-DOS環境のプログラミング方法については説明しません。 80386と8086とのプログラミング上の違いは、 386｜ASM 文法書か、付録Gの参考文献4と5を参照してください。MS-DOS環境のプログラミングについては参考文献3を参照してください。

本書では、16進数は、末尾に'h' を付けて表示し（たとえば、100h =10進数の256）、末尾に'h' が付いてない数字はすべて10進数です（たとえば、1024）。またLOW メモリ（従来のメモリ）という用語は、アドレスが0からfffff h 未満のメモリについて使用され、リアルモード実行時に直接アクセス可能なメモリを意味します。HIGHメモリ（拡張メモリ）という用語は、アドレスが100000h 以降のメモリを意味するのに使用されます。

第2章 DOS-Extenderの使い方

2.1 386ネイティブモードプログラムの構築

DOS-Extenderで実行するプログラムは、 386ネイティブモード用に構築する必要があります。そのためには、ソースコードは80386 をターゲットとしてコンパイルまたはアセンブルしなければなりません。オブジェクトファイルは、386 ネイティブモード用にリンクする必要があります。

高級言語のソースコードファイルは、80386 のオブジェクトコードを生成するコンパイラを使ってコンパイルしなければなりません。アセンブラのソースコードファイルは、80386 のオブジェクトコードを生成するために 386 | ASM アセンブラを使ってアセンブルする必要があります。たとえば、ファイル:HELLO.ASMを80386 用にアセンブルして、オブジェクトファイル:HELLO.OBJを作成するには次のように入力します。

```
386asm hello
```

オブジェクトファイルが作られたら、386 ネイティブモードで実行可能なファイルを作成するために 386 | LINKリンカによりリンクします。たとえば、ライブラリ:CLIB.LIB を使ってファイル:HELLO.OBJをリンクして、386 ネイティブモード実行可能ファイル:HELLO.EXP作成するには、次のように入力します。

```
386link hello -lib clib
```

上記の例で作成されたHELLO.EXP ファイルは、DOS-Extenderの386 ネイティブモードで実行することができます。DOS-Extenderは、 386 | LINKを使って作成されたREX形式 (386 ネイティブモードファイルの代替形式) のファイルを実行するのにも使用されます。オブジェクトファイルと実行可能ファイルの形式の詳細は、386 | LINK文法書を参照してください。

2.2 コマンドライン形式

DOS-Extenderは、ファイル:RUN386.EXE で提供されます。RUN386で使用されるコマンドラインの形式は、"RUN386"に続き、DOS-Extenderで処理されるコマンドラインスイッチ、実行する386 ネイティブモードプログラム名および386 ネイティブモードプログラムに渡されるコマンドラインパラメータから構成されます。386 ネイティブモードプログラムに対しファイル名拡張子の指定がなかった場合、デフォルトの拡張子".EXP"が使用されます。たとえば、コマンドラインパラメータを画面に表示する386 ネイティブモードプログラム:PECHO.EXPを実行するには、次のように入力します。

```
run386 pecho hello from protected mode!
```

プログラムをメモリ領域先頭1MBを超えたメモリに強制的にロードさせる場合、次のコマンドラインスイッチを使ってDOS-Extenderを起動します。

```
run386 -maxreal ffffh pecho Hello from above 1 MB!
```

2.3 コマンドラインスイッチ

コマンドラインスイッチは、DOS-Extenderのデフォルトの動作を変更するのに使用されます。デフォルトではDOS-Extenderは以下の動作を行います。

- MS-DOSおよびリアルモードのメモリ常駐プログラムが使用するメモリ以外をユーザアプリケーション用に割り当て、他のアプリケーションプログラム用のリアルモードメモリは残さない。
- 4KBをMS-DOSとBIOSのシステムコール用のデータバッファ用に割り当てる。
- 386ネイティブモードからリアルモードへの切換え時、1KBのバッファを4個スタックメモリ用に割り当てる。

コマンドラインスイッチの先頭は、マイナス符号"-"で始まり、そのあとにスイッチ名がつきます。各スイッチ名の形式は、完全形と短縮形の2形式があります。スイッチの引数は、スイッチ名の後に区切り記号として空白を1個入れたその直後に置きます。矛盾するスイッチがコマンドライン上にある場合、最後(右端)のスイッチが優先します。

引数として数字をとるコマンドラインスイッチがいくつかあります。デフォルトではこの数字は10進数(基数10)とみなされます。

16進数(基数16)を指定するには、数字の後に"h"か、"H"を付けます。次の二つの例は、スイッチ"-MAXREAL"の引数として同じ数値となります。

```
run386 -maxreal 200h hello
run386 -maxreal 512 hello
```

スイッチはコマンドラインだけからではなく、386 ネイティブモードプログラムに対して以下の三つのうちのいずれかの方法でも指定できます。

(1) プログラムのリンク時にいくつか指定する方法
(2) CFIG386 ユーティリティを使って、スイッチをDOS-Extenderのタスクイメージ(RUN386.EXE)に組込む方法
(3) プログラムの実行時、スイッチをコマンドラインに指定する方法

これらが複数設定された場合、リンク時の設定スイッチ、組込みスイッチ、コマンドラインスイッチの順に処理され、矛盾するスイッチが指定されている場合は、最後に処理されたスイッチが優先します。

2.3.1 LOW メモリスイッチ

-MINREALスイッチと-MAXREALスイッチは、DOS-Extenderが未使用領域として確保するリアルモードメモリ(LOWメモリ)の量を制御するのに使用されます。デフォルトでは、DOS-Extenderは使用可能なリアルモードメモリをすべてアプリケーションプログラム用に割り当てます。リアルモードメモリの使用方法については7.2.1 項を参照してください。

実行中のアプリケーションが別のプログラムを実行するためのシステムコールを行う場合、DOS-Extenderがその別プログラムの実行に必要なリアルモードメモリを十分に確保しているかを確認しなければなりません。-MINREALスイッチは、最低限確保しておくリアルモードのメモリ量を指定します。この最低限必要なメモリ量も確保できない場合、DOS-Extenderはプログラムを実行しません。

-MAXREALスイッチは、確保すべきリアルモードメモリの最大量を指定します。DOS-Extenderは最低限-MINREALメモリが確保されること、最大-MAXREALメモリまでをできるかぎり確保することを保証します。

-MINREALスイッチと-MAXREALスイッチは、引数として数値を指定します。この数値は、メモリサイズを16バイトパラグラフ単位(MS-DOSの標準とされるメモリ割当て単位)で指定します。引数は、65535 (FFFFh) 以下でなければなりません。デフォルトでは、DOS-Extenderは-MINREALと-MAXREALの両スイッチをゼロに設定します。これらのスイッチはプログラムリンク時にも指定できます。

形式： -MINREAL nparagraphs
-MAXREAL nparagraphs

省略形： -MINR nparagraphs
-MAXR nparagraphs

例： run386 -minreal 100h hello
run386 -minr 128 -maxr 512 hello

2.3.2　システムコールデータバッファスイッチ

-MINIBUFスイッチと-MAXIBUFスイッチは、MS-DOSとBIOSのシステムコールに使用されるデータバッファに割り当てるメモリ量を制御するのに使用されます。このバッファはファイルの入出力をする際に最も重要です。プログラムが一度に大量のデータを読み取ったり書き込んだりする場合に効率を良くするために大量のバッファを割り当てる必要があります。

-MINIBUFと-MAXIBUFの両スイッチは、引数に数字を指定します。この数は、バッファサイズを１KB単位で指定し、１から64までの任意の数字を指定します。デフォルトでは、DOS-Extenderは、-MINIBUFを１(KB)に、-MAXIBUFを64(KB)に設定します。DOS-Extenderが最低限必要な-MINIBUFで指定されたバッファを割当てできない場合、DOS-Extenderはプログラムを実行しません。割当て可能な場合、-MAXIBUFで指定されたサイズが割り当てられます。-MAXREALパラメータと-MAXIBUFパラメータの両方を満足させるメモリが不足する場合は、-MAXIBUFパラメータが優先します。このスイッチはプログラムリンク時にも指定できます。

```
形式：　　-MINIBUF nkilobytes
            -MAXIBUF nkilobytes

省略形：　-MINI nkilobytes
            -MAXI nkilobytes

例：　　　run386 -maxibuf 2 hello
            run386 -mini 64 filecopy
```

2.3.3　混合モードプログラムスイッチ

-REALBREAKスイッチと-CALLBUFS スイッチは、リアルモードコードと386 ネイティブモードコードの両方を持つプログラムのロード制御に使用されます。この事項は、第８章で詳しく説明します。

-REALBREAKスイッチは、リアルモードで実行、アクセスするためにLOW メモリにロードする必要のあるプログラムサイズを指定します。このスイッチは、プログラムのサイズを指定する引数をとります。このスイッチはリンク時にも指定できます。リンク時にこのスイッチを指定するのが便利なのは、引数がリアルモードのコードとデータの最後にあるパブリックシンボルの名前をとりうるときです。このスイッチが実行時に使用される場合、引数はリンクマップの情報から計算された絶対数でなければなりません。

-CALLBUFS スイッチはモード間呼出しバッファのサイズを指定します。モード間呼出しバッファは、アプリケーションプログラムがモード間プロシジャのコール時にデータバッファとして使用するために、LOW メモリ内に割り当てられます。バッファアドレスは、プログラム実行時にDOS-Extenderのシステムコールによって得られます。引数はバッファサイズ（単位：KB）で、64以下の値でなければなりません。デフォルトのバッファサイズは０です。このスイッチはリンク時にも指定できます。

形式：　　-REALBREAK nbytes
　　-CALLBUFS nkilobytes

省略形：　　-REALB nbytes
　　-CALLB nkilobytes

例：　　run386 -realbreak 200h -callb 2 switch
　　386link switch -realb END_REAL -callbufs 2

2.3.4　スタック割当てスイッチ

-NISTACKスイッチと-ISTKSIZE スイッチは、80386 を386 ネイティブモードからリアルモードに切換えるとき、スタック用のバッファに割り当てるメモリ量を制御するのに使用されます。スタックバッファの割当ての詳細は、第８章を参照してください。ほとんどのアプリケーションプログラムに対して、これらのパラメータはデフォルト設定値で問題ありません。

両スイッチとも引数として数値をとります。-NISTACKスイッチは、割り当てるスタックバッファの数を指定し、引数は４以上の数でなければなりません。-ISTKSIZE スイッチは、各スタックバッファのサイズ（単位：KB）を指定し、引数は１から64までの数でなければなりません。デフォルトでは、DOS-Extenderは、１KBのスタックバッファを４個割り当てます。これらのスイッチはプログラムリンク時にも指定できます。

形式：　　-NISTACK nbuffers
　　-ISTKSIZE nkilobytes

省略形：　　-NI nbuffers
　　-ISTK nkilobytes

例：　　run386 -ni 6 -istk 2 switch.exp

第3章 プログラム環境

3.1 プログラムの構造

386 ネイティブモードではセグメントの最大長は4GBで、セグメント内のオフセット長は、32ビットです。セグメントは、DOS-Extenderによって保持されるグローバルディスクリプタテーブル(GDT) とローカルディスクリプタテーブル(LDT) によって制御されます。セグメントレジスタにはセグメントセレクタと呼ばれる値がロードされます。セグントセレクタは16ビット値で、次の三つのフィールドを持ちます。

(1) GDTかLDTかを識別する1ビットのフィールド。

(2) 特権レベルセグメントへのアクセス制御に使われる2ビットのフィールド。

(3) 選択されたディスクリプタテーブル内のエントリをインデックスする13ビットのフィールド。

各ディスクリプタテーブルのエントリは8バイト長で、セグメントのリニアベースアドレス、セグメントサイズ、特権レベルおよびその他を保持します。詳細については、80386プログラマ用の文法書を参照してください。

DOS-Extenderは、自分自身のセグメントのマッピングにGDT を使い、アプリケーションプログラムのセグメントのマッピングにはLDT を使用します。プログラムは、常に特権レベル:0 (最高の特権レベル) で実行されます。そのため、ディスクリプタテーブルエントリとセグメントセレクタ値両方の特権ビットは常に0です。386 ネイティブモードで実行するためにプログラムがリンクされる時、リンカは、プログラムのコード、データおよびスタックセグメントのすべてを結合し、単一の大きなプログラムセグメントを構築します。セグメント化された386 ネイティブモードのリンクは現在サポートされていないので、プログラム中のセグメントセレクタ値を動的に得ることは不可能です。プログラムに最初に割当てられるセグメントはすべて、ハードウェア固定のセグメントセレクタ値を持ちます。

DOS-Extenderは、プログラムのロード時、このプログラムセグメントをマッピングする2つのエントリを LDTに作成します。セグメントセレクタ値000Ch によってアクセスされるディスクリプタは、読出し可能コードセグメントとしてプログラムセグメントをマッピングします。この値は、最初にCSレジスタにロードされるセレクタ値です。

セグメントセレクタ値0014h によってアクセスされるディスクリプタは、書込み可能デ ータセグメントとしてプログラムセグメントをマッピングします。この値は、DS、SS、E FSおよびGSの各セグメントレジスタに最初にロードされるセレクタ値です。

プログラムの最初の制御獲得時、レジスタには次のような値が格納されています。

DS = 0014h	EAX = 00000000h
ES = 0014h	EBX = 00000000h
FS = 0014h	ECX = 00000000h
GS = 0014h	EDX = 00000000h
CS = 000Ch	ESI = 00000000h
EIP = プログラムエントリアドレス	EDI = 00000000h
SS = 0014h	EBP = 00000000h
ESP = スタックの初期トップ	

3.2 プログラムセグメント

前節で簡単に説明したように386 ネイティブモードプログラムがメモリにロードされる時、ハードウェア固定のセグメントセレクタ値を持ついくつかのセグメントが、プログラム用のLDT 中に設定されます。プログラムの制御獲得時、LDT 中に設定されるセグメントを表3-1 に示します。

表３－１　ハードウェア固定のLDT セグメント

セグメントセレクタ値	説　　明
0004h	プログラム用のMS-DOSプログラムセグメントプレフィックス（PSP)を指す書込み可能データセグメント。
000Ch	プログラムセグメントを指す読出し可能コードセグメント。これは、CSレジスタに最初にロードされるセレクタ値である。
0014h	プログラムセグメントを指す書込み可能データセグメント。これは、DS、SS、ES、FSおよびGSの各レジスタに最初にロードされるセレクタ値である。
001Ch	VRAMを示す書込み可能データセグメント。これは、VRAMを直接アクセスしたいプログラムで使用される。
0024h	セレクタ値0004h でアクセスされるディスクリプタのコピーの書込み可能データセグメント（つまり、プログラムのPSP を指す）。
002Ch	プログラム用のMS-DOS環境ブロックを指す書込み可能データセグメント。環境ブロックには環境変数を識別する、末尾が０の一連のASCII文字列が入る。
0034h	MS-DOSの使用するメモリの下位の１MBすべてをマッピングする書込み可能データセグメント。

LDT（FMTOWNS用）

セグメント セレクタ値	説　　明
001Ch	TOWNS VRAM（16,32K色モード）
0104h	TOWNS VRAM（16,32K色モード）
010Ch	TOWNS VRAM（256色モード）
0114h	スプライト　パターンRAM

これらのハードウェア固定のセグメントのほかに、新しいセグメントがメモリ割当てシステムコールを使ってLDT 内に作成されます。

3.3　プログラムセグメントプレフィックス（PSP）

MS-DOS上で実行するプログラムはすべてプログラムセグメントプレフィックス（PSP)を持ちます。PSP は256 バイトのメモリブロックで、この中にはMS-DOSによって維持される情報と実行開始時にプログラムに渡されるコマンドラインが格納されます。PSP はMS-DOSでアクセス可能である必要があり、常に下位１MB内の空間に置かれます。セグメントセレクタ0004h と0024h は、386 ネイティブモードプログラムのPSP をマッピングするために設定されます。

コマンドラインテール（プログラムを起動したコマンドライン上にあるプログラム名の後の文字）は、PSP 内のオフセット0081h にあります。PSP 内のオフセット0080h のバイトはコマンドテール文字列の長さを示します。この文字列の長さには、コマンドテールを終了させるキャリッジリターン文字は含まれません。

PSP 内のオフセット0080h の128 バイトのバッファは、初期ディスク転送領域（DTA)としても設定されます。このバッファはMS-DOSディレクトリ検索システムコールによって使用されます。

PSP の先頭128 バイトには、プログラムによってオープンされたファイルのファイルハンドルを含むMS-DOSによって維持される情報が格納されています。PSP のこの部分の詳細は、参考文献３を参照してください。DOS-ExtenderはMS-DOS情報のほかに、プログラムサイズに関する情報をPSP に格納します。DOS-ExtenderがPSP に格納する情報を表3-2 に示します。

表3-2　DOS-Extender PSPフィールド

PSP オフセット	格　納　情　報
0000005Ch	最小限必要なプログラムセグメントのサイズ（単位：バイト）。これは.EXPファイルから実際に読込まれたバイト数と、プログラムに最小限割り当てる必要のあった割増しのデータの量とプログラムのベースオフセットリロケーション値(3.6節参照）とを加えた値と等価である（最小限必要な割増しデータは、未初期化データ用とスタックメモリ用に使用される）。
00000060h	割り当てられたプログラムセグメントのサイズ（単位：バイト）。これは、ロード時、実際にプログラムに割当てられたメモリ量を示し、プログラムセグメントのリミット＋1と等価である。
00000064h	ロード時、プログラムセグメントに割当てられる割増しメモリの全量（単位：バイト）。割り当てられたプログラムセグメントサイズからこの値を引いた値が、.EXPファイルから読込まれたデータ量プログラムの、ベースオフセットリロケーション値を加えたものとなる。

3.4　環境ブロック

MS-DOS上で実行するプログラムは、すべて環境ブロックを持っています。環境ブロックには"name=string" の形式の末尾が00h で終了する一連の文字列が含まれています。ブロック全体は、00h をさらに続けて終了します。環境ブロックはMS-DOSによって使用されるので常に下位1MB以内のメモリ内におかれます。セグメントセレクタ002Ch は、プログラムの環境ブロックをアクセスするために使用されます。そのほかに、環境ブロックのリアルモードパラグラフアドレスがオフセット 002EhでPSP に格納されます。

環境ブロックにはシステムのコマンドインタプリタ（通常、COMMAND.COM)によって使用される文字列が格納されています。このブロックは、情報を親プログラムから子プログラムに受渡しするのにも使用することができます。詳細は参考文献3を参照してください。

3.5　メモリの割当てと割当て解除

386 ネイティブモードプログラムは、標準MS-DOSシステムコールを使ってメモリの割当てと解放を行う機能を持ちます。しかし、通常のMS-DOSシステムコールと異なり、DOS-Extenderは、メモリ割当てコールをMS-DOSに渡さずに、DOS-Extender自身で4KBページを単位としたメモリの割当てとその解放を行います。さらに、"allocate memory" システムコールは、割り当てられたメモリをマッピングするために、LDT 内のセグメントディスクリプタエントリをLDT に作成します。

DOS-Extenderは、MS-DOSメモリの割当て／解放ファンクションのコールを行う機能も持ちます。このコールはLOW メモリから必要なメモリを割り当てるためにサポートされています。これらのコールは、同一プログラム中にリアルモードと386 ネイティブモードのコードが混在しているプログラムで必要とされます。これについては第８章で詳しく説明します。

システムコールを使ってメモリの割当て／解放を行うプログラムは、プログラムのロード時、プログラムに割り当てられたメモリ量を知る必要があります。プログラムをリンクする時.EXPファイル中にロード時に割り当てる割増しメモリの最大量、最小量を指定する２個のパラメータが初期設定されます。デフォルトでは、リンカは、割り当てる割増しデータの最小量を未初期化データとプログラムの使用するスタックメモリの量との合計に設定し、最大量をマシン内の全メモリに設定します。リンカの-MINDATAスイッチと-MAXDATAスイッチは、これらのデフォルトを変更するのに使用できます。素性のよいプログラムにするには、不要なメモリを解放するためシステムコールを行うかすべてを-MAXDATAスイッチを用いてリンクしてプログラムロード時に、ローダが使用可能なメモリのすべてをそのプログラムに割り当てないようにしてください。これを実行しない場合、システムコールによってメモリの割り当て（および、別のプログラムのEXEC）をすることは不可能です。その理由は、プログラムが、既にマシン内の全メモリを占有しているからです。

3.6　ヌルポインタの検出

プログラムエラーの一つに、ヌル（ゼロ）ポインタを使ってメモリを参照しようとすることがあります。

DOS-Extenderは、このエラーを自動的に検出します。プログラムのリンク時、-OFFSETスイッチを使って、プログラムの先頭をプログラムセグメントのベースより１ページ（４KB／ページ）以上の上位に置くことができます。DOS-Extenderは、プログラムをロードしてプログラムセグメントを初期化する時、セグメントの始めの未使用ページをマッピングしません（ページングの詳細は、第７章を参照してください）。したがって、プログラムセグメント内のマッピングされていない領域内のポインタで参照しようとすると、プログラムのコードやデータを汚すことなくページフォルトが発生します。

未使用ページはメモリにマッピングされないので、このオプションを使用しても、メモリを浪費することはありません。

3.7　別プログラムに対する EXEC の実行

標準MS-DOSシステムコール（拡張パラメータあり）で、別プログラムに対してEXECを実行することができます。子プログラムの終了時、制御権は親プログラムに戻ります。EXECシステムコールの詳しい形式は、付録Ｄを参照してください。

別プログラムを実行するには、プログラムのロードに必要な空きメモリが十分なければなりません。

子プログラムが標準MS-DOSリアルモードプログラムの場合、プログラムロードに必要なLOW メモリが十分空いていなければなりません。子プログラムが386 ネイティブモードプログラムの場合、DOS-Extenderの2番目のコピーをロードするのに十分なLOW メモリ（約100KB)がなければなりません。その上、386 ネイティブモードプログラム自体をロードするのに十分な追加メモリ(LOWメモリかHIGHメモリ) も必要です。

デフォルトでは、DOS-Extenderは、アプリケーションプログラムのメモリを割り当てる時、4KBページ単位で、LOW メモリをすべて割り当てます。このデフォルト値は、プログラムリンク時か、DOS-Extenderによるプログラム実行時に、-MINREALと-MAXREALのコマンドラインスイッチを使用して変更することができます。これらのスイッチは、別プログラムを実行させるためには使用しなければなりません。

別の386 ネイティブモードプログラムを実行する場合は、使用可能なHIGHメモリ量も考慮しておく必要があります。その理由は、386 ネイティブモードプログラムは、LOW メモリだけでなく、HIGHメモリも使用できるからです。HIGHメモリは、プログラムがメモリ割当てシステムコールを行うと、動的に割り当てられます（3.5 節参照）。したがって、別の386 ネイティブモードをEXEC（実行）するプログラムには、子プログラムにメモリを残しておくため、実際に必要なメモリだけを割り当てる必要があります。メモリ割当ての詳細は、第7章を参照してください。

3.8 環境検出

386 ネイティブモードプログラムは、標準MS-DOS環境をDOS-Extender上で実行可能です。プログラムの実行環境は、INT 21h ファンクション30h (get DOS version number)システムコール後に EAXレジスタの上位ワードを調べることによって決定されます。このシステムコールは、付録Dで説明します。

アプリケーションプログラムからみると、これらの動作環境間にはハードウェア割込みが引きつがれることを期待することを除いて全く違いがありません。

第4章 ハードウェアアクセス

4.1 特権レベル

80386プロセッサはプログラムを特権レベル0から3（0：最高特権レベル）で実行します。プロセッサフラグワードには入出力特権レベルフィールドがあります。現特権レベルが入出力特権レベルより番号が大きい(低い特権レベルの)場合、IN命令とOUT 命令（ハードウェア入出力ポートのアクセス用）割込み有効命令および割込み無効命令は、一般に保護プロセッサ例外を引き起こします。

さらに、ゼロ以外の特権レベルでの実行時、使用を許されないプロセッサレジスタのロード命令とストア命令も一般保護例外を引き起こします。

DOS-Extender上の386ネイティブモードプログラムは、常に特権レベル0で実行されます。したがって、これらのプログラムの実行制限はなく、DOS-Extenderが特権命令をシミュレートすることによって、オーバヘッドを持ち込む必要はありません。

4.2 ハードウェア入出力

386ネイティブモードのハードウェア入出力は、リアルモードの場合と同じ方法で行われます。すなわち、IN命令とOUT 命令とを使って、入出力ポートの読出しと書込みを行います。メモリマップドハードウェアデバイスに対しては、システムコールを使用して80386プロセッサの4GBアドレス空間の任意の場所に物理メモリをマッピングするLDT セグメントを作ることができます。DOS-Extenderシステムコールの一覧表を付録Eに示します。メモリの下位の1MB内のメモリマップドデバイスには、セグメントセレクタ0034hでアクセスできます。このセグメントセレクタは、物理メモリの下位1MB全体をマッピングします。

4.3 画面アクセス

DOS-Extenderは、セグメントセレクタ001Chを自動的に設定して、VRAMをマッピングします。性能上の理由でVRAMを直接読み出したり書き込んだりしたいプログラムは、セグメント001Chを使えばその読み書きが可能となります。

4.4　80387 浮動小数点コプロセッサのプログラミング

80386システムでは、80387命令実行時の80386 の動作モードによって決定されます。コプロセッサ通信が正しく行われ、常にその結果が期待通りの形式で格納されることを保証するために、DOS-Extenderは、コプロセッサが現命令の実行を終了したことを確認するために、モードの切換前に必ずFWAIT 命令を実行します。

FSAVE、FRSTOR、FSTENVおよびFLDENVの命令(または、80386または80387の他の命令) を含むプログラムのアセンブル時、命令のオペランドサイズ属性は、以下の規則によって決定されます。

1. 現セグメントの使用型がUSE16の場合、デフォルトのオペランドサイズは16ビットです。その使用型がUSE32の場合、デフォルテのオペランドサイズは32ビットです。リアルモードで実行中のプログラムは、USE16セグメントでの実行を定義されます。DOS-Extenderの設定する386ネイティブモードプログラムセグメントは、常にUSE32セグメントです。
2. オペランドサイズオーバライドプレフィックスバイト(66h)が命令opコードの前にある場合、その命令の実際のオペランドサイズ属性は切り換えられます。

このように、どちらのオペランドサイズ属性も、リアルモードプログラム、386ネイティブモードプログラムの両方に使用できます。特に指定のない限り、オペランドサイズ属性はリアルモードプログラムでは16ビットとなり、386ネイティブモードプログラムでは32ビットとなります。オペランドサイズオーバライドバイトとアセンブラによるそれらの生成条件の詳細は、386 | ASM 文法書を参照してください。

386ネイティブモードプログラムが80387コプロセッサを使用する場合に問題があります。それは、D0ステップ以前の80386を使用した時、80837 4プロセッサのやりとりでマシンをハングアップすることがあります。この問題を回避する方法として、次のようなものがあります。

(1) 浮動小数点演算を80387を使わずにソフトウェアでシミュレートする。

(2) D0またはそれ以後のステップの80386を使う。

(3) このバクが発生した時にハードウェアに必要な機能のページング機構を使わない。これは、"-NOPAGE"スイッチを指定することで可能となります。

第5章 システムコール

5.1 MS-DOSシステムコール

DOS-Extender上では、386 ネイティブモードプログラムはリアルモードプログラムと全く同じようにMS-DOSシステムコールはソフトウェア割込みを使います。DOS-Extenderはソフトウェア割込みを検出すると、必要に応じて関数パラメータの変更を行った後、リアルモードに切換えて割込みを再発行してMS-DOSを呼び出します。通常のシステムコールに対する呼出しシーケンスは、リアルモードの場合と全く同じです。メモリ内のデータか、バッファが示すポインタを渡すシステムコールの場合、リアルモードではポインタのオフセット部の幅が16ビットであるのに対し、386 ネイティブモードでは32ビットです。これは、リアルモードセグメントは64KBに制限されているのに対し、386 ネイティブモードセグメントの最大サイズは 4 GBであるからです。

旧式のファイル制御ブロック（FCB)のファイルI/O コールは、DOS-Extenderではサポートされないので、代わりにハンドルベースファイルI/O コールを使用しなければなりません。また、MS-DOSの割込みベクタの入手と設定ファンクションはサポートされません。それらはDOS-Extenderシステムコールに置き換えられ、２つのプロセッサモードによって増えた複雑さを処理します。割込みについての詳細は第６章を参照してください。

マニュアルに記載されませんがプログラマによって使用されるMS-DOSシステムコールは数多くあります。マニュアルに記載されてないシステムコールは、レジスタ内容を変更しない状態で渡されます。これによって、すべてのファンクション（ただし、値をセグメントレジスタに渡す必要のあるシステムコールは除く）はリアルモードの場合と同様に機能します。各々のMS-DOSシステムコールの詳細は、付録Ｄを参照してください。

5.2 386 | DOS-Extenderシステムコール

DOS-ExtenderシステムコールはMS-DOSシステムコールと同様に、ソフトウェア割込みにより呼び出されます。システムコールは、割込み21h で行われます。この時レジスタAHは25h に設定され、希望するシステムコールの関数コードはレジスタAL中に設定されます。各々のDOS-Extenderシステムコールの詳細は付録Ｅを参照してください。

5.3 システムコール時のスタックの使用

DOS-Extenderはシステムコールが行われる時に、ユーザスタックの内 250バイトを使用します。すべてのプログラムはシステムコールを行う時に、少なくとも 250バイト以上のスタックを用意しなくてはなりません。

第6章 割込みと例外

6.1 386｜DOS-Extender標準割込み処理

割込みは、次の三つに分類できます。

(1) ハードウェア割込み（外部ハードウェアにより発生）
(2) ソフトウェア割込み（INT 命令の実行により発生）
(3) プロセッサ例外（80386 プロセッサがプログラミングエラーを検出した時に発生）

上記三つの割り込みはDOS-Extenderにより同じように処理されます。この章では、386 ネイティブモードで割込みが発生した時の処理について説明します。また各説では、各割込み処理で、標準処理と異なる処理について説明します。

割込みが386 ネイティブモードで発生すると、アプリケーションが割込みベクタをインタセプトした場合を除いて、DOS-Extenderは必ず制御権を獲得します。制御権を獲得した後DOS-Extenderは、プロセッサをリアルモードに切換え、割込みをソフトウェア割込みとして再発行します。すべての汎用レジスタとプロセッサフラグの値は、リアルモード割込みハンドラへのモード切換えに渡って保存されます。SS:SP レジスタは、DOS-Extenderによって内部的に維持されるスタックメモリバッファを指すように設定されます。スタックバッファデフォルト時のサイズは1KBです。スタックバッファはプログラム実行時-ISTKSIZE コマンドラインスイッチによって制御されます。CS:IP レジスタは、リアルモードでの割込み再発行時は、DOS-Extenderコード内を指します。他のセグメントレジスタ（DS、ES、FS、GS）の内容は、リアルモードハンドラの制御獲得時に不定となります。リアルモード割込みハンドラの処理終了時、DOS-Extenderはモードを再び386 ネイティブモードに切換え、割込み発生時に実行していた386 ネイティブモードプログラムに戻ります。386 ネイティブモードプログラムが制御権を得る時にリアルモード割込みハンドラの残した値が汎用レジスタとフラグすべてに入っています。当然、SS:ESPレジスタとCS:EIPレジスタの内容は、386 ネイティブモードにおける割込み発生時の値に戻されます。DS、ES、FSおよびGSのレジスタ内容も元の値に戻されます。

DOS-Extenderをいったん386 ネイティブモードからリアルモードに切換え、再び386 ネイティブモードに戻すのに必要な総オーバヘッドは、約200 μsです。

6.2 ソフトウェア割込み

ソフトウェア割込みが386 ネイティブモードで発生すると、DOS-Extenderはプログラムをリアルモードに切換え、割込みを再発行します。ソフトウェア割込みがMS-DOSまたはBIOSシステムコールの場合、DOS-Extenderは必要に応じ、まず初めにパラメータを32ビットの386 ネイティブモード形式から、DOS の16ビットリアルモード形式に変換します。MS-DOSシステムコールの32ビットの386 ネイティブモード形式については、付録Dで説明します。

パラメータとしてポインタを持たないシステムコールは、通常はなにもしないでそのままリアルモードに渡されます。すなわち、6.1 節で述べた標準割込み処理が実行されます。ポインタを伴うシステムコールは、変更の必要があります。その理由は、セグメントレジスタの内容は、386 ネイティブモードではGDT かLDT のエントリへのインデクスであり、リアルモードではメモリパラグラフアドレスであるためです。さらに386 ネイティブモードデータバッファは、物理メモリの1MBより上位のアドレスに位置することがあります。その場合リアルモードではそのバッファにアドレス指定することが不可能です。従ってポインタパラメータ付きのシステムコールが実行される場合、ポインタの一部に使用されるセグメントレジスタの内容はリアルモードハンドラの制御獲得時に破壊されることはありません。しかし、そのセグメントレジスタの内容は386 ネイティブモードの場合と同じではなく、適当なリアルモードの値に変換されています。

ポインタを伴うシステムコールが実行されると、DOS-Extenderはポインタを386 ネイティブモードからリアルモードのポインタに変換するか、またはデータを内部バッファに格納します。データサイズがある一定の値（現在は1KB）を超えない限り、データは必ずバッファに入れられます。その理由は、ポインタ変換のオーバヘッドの方が、データをバッファに入れるオーバヘッドより大きいからです。大量のデータに対しては、データバッファがアドレスより下位にある場合、DOS-Extenderは386 ネイティブモードポインタを等価なリアルモードポインタに変換します。システムコール時のデータ格納用の内部バッファのサイズは、1KBから64KBの範囲で、-MINIBUFと-MAXIBUFのコマンドラインスイッチによって制御されます。格納するデータ量が内部バッファより大きい場合、DOS-Extenderは、すべてのデータの転送が終了するまでMS-DOSまたはBIOSシステムコールを繰り返し実行します。システムコールの復帰時、標準割込み処理のところで述べたように、通常システムコール前のセグメントレジスタ値が復元されます。これに関する例外はただ一つ、システムコールが結果の一つとしてポインタを返す場合です。この場合、DOS-Extenderは、このリアルモードポインタを386 ネイティブモードポインタに変換します。この変換は通常、セグメント0034h （物理メモリの下位の1MB全体をマッピングするセグメント）を使います。

DOS-Extenderは、上記の方法でシステムコールの拡張をマニュアルに記載されたMS-DOSシステムコールと標準BIOSシステムコールに対してだけ行いました。マニュアルに記載されないすべてのシステムコール、つまりポインタを使用する他のプログラム（たとえばネットワークドライバ）に対するシステムコールは、アプリケーションプログラムによって扱われなくてはなりません。

DOS-Extenderは、指定された場合にはセグメントレジスタを含む、すべてのリアルモードレジスタの値を伴う、任意のリアルモード割込みを発行するためのシステムコール(付録Eを参照) を提供します。このシステムコールは、リアルモード割込みハンドラによってセグメントレジスタに残されたリアルモードの値も返します。このDOS-Extenderシステムコールは、ソフトウェア割込みで実現されるリアルモードシステムコールを作るために使うことができます。第8章では, リアルモードプログラムとやりとりする386 ネイティブモードプログラムの作成方法についての一般的な問題を説明します。

6.3 ハードウェア割込み

DOS-Extenderは、ハードウェア割込みに対して常に標準割込み処理 (6.1 節参照) を行います。この処理では、リアルモード割込みハンドラは汎用レジスタとプロセッサフラグの内容を変更することができますが、この変更はハードウェア割込みハンドラが絶対にしてはいけないことであることに注意してください。したがって、DOS-Extenderではリアルモードでそうであるように、誤りのあるハードウェア割込みハンドラはシステムを破壊してしまいます。

6.4 プロセッサ例外

プロセッサ例外が386 ネイティブモードで発生すると、DOS-Extenderはリアルモードに切換え、プロセッサ例外をソフトウェア割込みとして再発行します。プロセッサ例外は、通常アプリケーションプログラムのバグであることを示します。したがって、DOS-Extenderはすべてのプロセッサ例外 (リアルモードでのデバッグ例外を除く) を引きつぎます。例外の発生時、DOS-Extenderはプログラムを終了させ、例外を識別するメッセージと例外発生の原因を作った命令のアドレスを示します。

プロセッサ例外が原因でプログラムが終了した場合、そのプログラムは、Phar Lap社の386 | DEBUG でデバッグする必要があります。デバッガは例外の発生時に制御権を獲得し、例外の原因を見つけるためユーザがレジスタ内容とメモリ内容を調べられるようにします。

6.5 割込みのインタセプト

割込みベクタの保存と変更に使用されるDOS-Extenderシステムコールにはいくつかあります。これらのシステムコールは、標準的なMS-DOS割込みベクタ設定を置き換えるものです。DOS-Extenderシステムコールの詳細は、付録Eで説明します。

割込みのインタセプトは、リアルモードプログラムの場合より386 ネイティブモードプログラムの場合の方が複雑です。その理由は、386 ネイティブモードプログラムは、386 ネイティブモードだけでなくリアルモードで発生する割込みも処理できるようにしておく必要があるからです。割込みが386 ネイティブモードで発生すると、ハンドラが割込みディスクリプタテーブル(IDT) を介して呼び出されます。IDT は、メモリ内ならどこでも常駐できます。

またIDT は256 の割込み各々に対して8バイトのディスクリプタを持ちます。ディスクリプタにはディスクリプタの型を識別するゲートタイプコードの他に、割込みハンドラルーチンのエントリポイント（セグメントセレクタとオフセット）が含まれます。386 プロセッサではIDT 中にタスクゲート、割込みゲートおよびトラップゲートを使用できます。

DOS-Extenderは、全IDT ディスクリプタエントリに対して必ず割込みゲートを使用します。これは、ハンドラの制御獲得時に割込みが必ず無効とされ、ハンドラはタスク切換えなしに直接実行されることを意味します。

割込みがリアルモードで発生すると、386 プロセッサはアドレステーブル（メモリの最初の1MB内ならどこでも置くことができますが、MS-DOS上での実行時は必ず00000h－003FFhの間に置かれます）を介して呼び出します。各テーブルのエントリには、呼び出される割込みハンドラの4バイトリアルモードアドレス（セグメント：オフセット）が登録されています。割込みハンドラがリアルモード制御を獲得すると、割込みは必ず無効とされます。　割込みをインタセプトする386 ネイティブモードプログラムは、割込み発生時のプロセッサ動作モードを知っていなければなりません。ハードウェア割込みは、いつでも発生させることができるので、ハードウェア割込みをインタセプトするプログラムは、386 ネイティブモードだけでなくリアルモードでも割込みの発生の可能性があることを考慮する必要があります。言うまでもなく、ソフトウェア割込みはプログラムが割込みを発行した時だけ発生します。したがって、どのプログラムがどのソフトウェア割込みを使用するかを知っていれば、割込み発生時のプロセッサ実行モードがわかります。プロセッサ例外は、一般にプログラムにバグがあることを示すので、通常アプリケーションプログラムでインタセプトされることはありません。例外はリアルモードでも386 ネイティブモードでも発生しますが、純粋の386 ネイティブモードプログラムでは、プロセッサ例外を386 ネイティブモードでのみ発生させます。リアルモード実行コードと386 ネイティブモード実行コードの両方を持つプログラムは、当然どちらのモードでもプロセッサ例外が発生します。

386 ネイティブモード割込みベクタ設定システムコール（ファンクション 2504h）は、386 ネイティブモード割込みハンドラを設定するために、IDT だけを変更します。この変更は386 ネイティブモードでしか発生しないことをプログラムが認識しているソフトウェア割込みとプロセッサ例外をトラップする場合に役立ちます。

リアルモード割込みのベクタ設定システムコール(ファンクション2505h)は、リアルモード割込みハンドラを設定するために、リアルモード割込みベクタテーブルだけを変更します。DOS-Extenderは、386 ネイティブモードでの割込み発生時に、必ずリアルモードに切換えて割込みを再発行します。このため、リアルモード割込みベクタのインタセプトは、割込み発生時のモードとは関係なくリアルモードハンドラが制御権を獲得することを意味します。このファンクションはリアルモード処理のハードウェア割込みをインタセプトする場合か、リアルモードで発行し、リアルモードで処理する方が容易なソフトウェア割込み（MS-DOS重大エラー割込みなど）をインタセプトする場合に役立ちます。

386 ネイティブモードで制御を得るための割込み設定システムコール(ファンクション2506h)は、386 ネイティブモード割込みハンドラを設定するためにIDT を変更し、リアルモード割込みベクタテーブルを変更して386 ネイティブモードへの切換えと割込みの再発行を行うルーチンを指すようにします。したがって、このシステムコールによる割込みのインタセプトは、386 ネイティブモード割込みハンドラに、最初の割込み発生時のモードとは関係なく、制御権を与えます。このファンクションは386 ネイティブモードで処理するためにハードウェア割込みをインタセプトする場合か、リアルモードで発生したプログラムが386 ネイティブモードで処理を行うように望むソフトウェア割込みをインタセプトする場合に役立ちます。リアルモードから386 ネイティブモードに切換えを経て制御権を獲得する割込みハンドラに対して与えられる環境については、6.5.2 項を参照してください。

リアルモードと386 ネイティブモードの割込みベクタ設定システムコール(ファンクション2507h)は、386 ネイティブモード割込みハンドラを設定するためにIDT を変更し、リアルモード割込みハンドラを設定するためにリアルモード割込みベクタテーブルを変更します。このシステムコールと、ファンクション 2504hと 2505hの 2 つの別れたシステムコールを行うこととの違いは、このファンクションは両ベクタを不可分に設定すること、つまり両ベクタの変更が終了して初めて割込みが有効になることです。これは、例えばリアルモードと386 ネイティブモードの別々のハンドラを使ってハードウェア割込みをインタセプトする場合に重要となります。

6.5.1 割込みハンドラの作成

割込みが発生した場合、プロセッサはプロセッサフラグと現在のCS:(E)IP命令アドレスをスタックにプッシュした後、制御を割込みハンドラに移します。割込みハンドラは変更するレジスタ内容をすべて保存し、処理を実行し、保存したレジスタ内容をすべて復元し、最後に割込み復帰命令で復帰（保存されていたフラグと命令アドレスをスタックから復元する）します。

割込み復帰命令には、IRET命令ニーモニックをアセンブルしてつくられる16ビット形式のIRET命令と、IRETD 命令ニーモニックをアセンブルしてつくられる32ビット形式のIRETD 命令の 2 形式があります。正しい命令を選択することはきわめて重要です。割込みが386 ネイティブモードで発生すると、プロセッサはセグメント使用型属性などとは関係なく、必ず32ビット形式で情報をスタックに保存します。したがって、386 ネイティブモード割込みハンドラは、必ずIRETD 命令を使って復帰しなければなりません。割込みがリアルモードで発生した場合は、プロセッサは、必ず16ビット形式で情報をスタックに保存します。したがって、リアルモード割込みハンドラは、必ずIRET命令を使って復帰しなければなりません。割込み発生時にプロセッサが実行する処理の詳細は参考文献 1 を参照してください。

ハードウェア割込みはいつでも発生することがありうるので、ハードウェア割込みハンドラは、変更するレジスタ内容をすべて保存／復元しなければなりません。ソフトウェア割込みハンドラは、システムコール（例えばMS-DOSコール）をインプリメントするのによく使用されるため、ソフトウェア割込みハンドラはパラメータをシステムコールから返す

ためにレジスタ内容を変更することがよくあります。割込みハンドラでプロセッサフラグを変更したい場合には、スタック上に退避されたフラグを変更する必要があります。なぜなら、これらのフラグは割込み復帰命令によって復元されるからです。

ハードウェア割込みはいつでも発生することがあるので、ハードウェア割込みハンドラは自分の環境を仮定することはできません。100 バイト以上のスタックを使用するハンドラは、制御を受け取る時にスタックの変更をしなくてはなりません。CSレジスタを除くセグメントレジスタの値の仮定をしてはいけません。もし割込みハンドラが高レベル言語で書かれたルーチンの呼出しを行っている時は、ハンドラはその言語に必要な環境のセットアップをしなければなりません。ほとんどの言語では、DSとESがデータセグメントを指し（セグメントセレクタ:0014h）、SSレジスタがDSと同一のセグメントを指し（ハードウェア割込みハンドラが制御を受け取った時は除く）、そしてEFLAGSレジスタのディレクションフラグがクリアされている必要があります。またハードウェア割込みハンドラは再び入られることに注意しなくてはなりません。このハンドラは、リエントラントに作成するか、または次の項目を守って作成する必要があります。

(1) 残りの割込みを無効にする。

(2) 8259割込みに対しその処理が終了するまでEOI(End Of Interrupt) を発行しない。

(3) ハンドラのゲート入口にセマフォを使う。

最後に、ハードウェア割込みハンドラは、MS-DOSまたはBIOSシステムコールを使ってはいけません。理由はMS-DOSとBIOSどちらともリエントラントでないからです。

既存のリアルモード割込みハンドラを、386 ネイティブモードで実行するために変更する場合、または既存のリアルモードハードウェアデバイスドライバを使用するプログラムを作成する場合、レジスタ内容の保存に注意を払うことが特に重要です。この良い例がMicrosoft のマウスドライバです。マウスドライバの典型的な使用方法は、マウスを移動するたびまたはマウスボタンを押すたびに制御権を獲得する割込みハンドラを設定するのにファンクションコールをすることです。マウスドライバは、マウス割込みの発生のたびに、16ビットレジスタの内容をすべて保存してから、指定のハンドラを呼び出します。したがって、マウスを使用する8086標準プログラムはレジスタ内容の保存について心配する必要はありません。ただし、80386 プログラムがマウスドライバを使用する場合は別です。その理由は、これらのプログラムは 80386の32ビットレジスタの内容を変更する能力を持っているからです。このようなことは、特に386 ネイティブモード割込みハンドラがDOS-Extenderシステムファンクション2506h を使って設定された場合に発生しやすくなります。

6.5.2 割込みを使ったモードの切換え

アプリケーションプログラムがソフトウェア割込みを使ってモードの切換えを自由にできることは、既に説明しました。この割込みの方法は二つあります。一つは386 ネイティブモードからリアルモードに切換えるのに使用されます。もう一つは、リアルモードから386 ネイティブモードに切換えるのに使用されます。リアルモード割込みハンドラはシステムファンクション 2505hを使って設定されます。また、386 ネイティブモード割込みハンドラはシステムファンクション 2506hを使って設定されます。

386 ネイティブモードで実行しているプログラムは、前者の割込みを発行するだけでリアルモードに切換わることができます。割込みが発生すると、リアルモード割込みハンドラは、汎用レジスタの内容は変わることなく制御権を獲得しますが、この時セグメントレジスタの内容は破壊され、一時スタックに退避されます。詳細は6.1 節を参照してください。プログラムがリアルモードで実行されている場合、プログラムが後者の割込みを発行することによって386 ネイティブモードに切換えることができます。割込みが発生すると、386 ネイティブモード割込みハンドラは、すべての汎用レジスタとプロセッサフラグの内容をどれも変更しないで制御権を獲得します。SS:ESPレジスタが、リアルモードで使用されていたスタックをどれも使用するように設定されます。この設定は、SSをGDT 内の物理メモリの最初の1MBをマッピングするセグメントセレクタに設定し、ESP をリアルモードスタックの物理アドレスに設定することによって行います。DS、ES、FS、GSの各レジスタは、初期設定時にプログラムが最初に制御権を獲得した時と全く同じように設定されます。

リアルモードと386 ネイティブモードコードの両方を持つプログラムの作成方法は、第8章で詳しく説明します。

第7章 メモリ管理

7.1 DOS-Extender メモリモデル

DOS-Extenderはメモリ管理を実行するために、80386 プロセッサに組込まれているページング機能とセグメンテーション機能の両方を使用します。

386 ネイティブモードセグメントはセグメントセレクタと呼ばれる値によって参照されます。セグメントセレクタはセグメントレジスタにロードされ、GDT またはLDT(第3章を参照) 内のディスクリプタエントリを参照します。セグメントディスクリプタにはセグメントの32ビットリニアベースアドレスとセグメントリミットが格納されています。セグメントリミットを超えてデータを参照しようとすると、結果としてプロセッサ例外が発生します。

80386 ページングハードウェアは、32ビットリニアベースアドレスをアドレスバス上に出力される32ビット物理アドレスに変換します。リニアアドレスは、メモリの物理ページ選択用の20ビットと選択されたメモリページ内のオフセットを示す12ビットから構成されます。したがって、物理メモリページのサイズは4KBです。80386 のページングとセグメンテーションの詳細は、参考文献1を参照してください。80386 におけるメモリ参照には、必ずセグメントセレクタ値（暗黙に指定されるか、直接命令で指定されるかのどちらか）と、セグメント内のオフセットが含まれます。386 ネイティブモードでメモリ参照が行われる場合、プロセッサは次の方法で物理メモリアドレスを構築します。

1. プロセッサは参照されたセグメントのリニアベースアドレスを、LDT またはGDT のセグメントディスクリプタから得る。
2. プロセッサはセグメント内のオフセットを、セグメントベースアドレスに加えて、80386 の4GBアドレス空間のリニアアドレスを作成します。
3. プロセッサに内蔵されたページング機能を使って、リニアアドレスをアドレスバス上に出力される物理メモリアドレスに変換します。

386 ネイティブモードアプリケーションプログラムは通常、この全過程にかかわりません。本章は、システムディスクリプタテーブルまたはページテーブルを使用／操作する必要のあるプログラム用の参考情報を提供するために設けられました。

7.1.1 ページング

ページングはDOS-Extenderの設定するページテーブルを通して 80386に制御されます。全ページテーブルのサイズは４KBで、物理ページ１ページをマッピングする４バイトのエントリを1024個持っています。各々のエントリには、20ビットページフレームアドレス（物理ページのサイズは４KBなので、４GBアドレス空間全体を指定するのにはこれで十分）とフラグの12ビットが格納されます。

ページディレクトリと呼ぶ１ページのテーブルがシステム内にあります。80386 システムレジスタCR3は、ページディレクトリの物理アドレスを保持します。ページディレクトリの各エントリは、ページテーブルとして使用される物理メモリページをマッピングします。したがって最大1024のページテーブルがシステム内に存在し得ます。ページテーブルの各エントリは物理メモリページをマッピングします。

386 ネイティブモードの32ビットリニアアドレスは三つの部分から構成されます。上位10ビットはページディレクトリ内の一つのエントリを選択します。このエントリは特定のページテーブルを指しています。次の10ビットはそのページテーブル内の一つのエントリを選択します。このエントリには特定の物理メモリページのポインタが入っています。下位12ビットは、選択された物理メモリページ内のオフセットです。処理の詳細は参考文献１を参照してください。メモリ管理を行うために、DOS-Extenderは、セグメント内のオフセットが物理メモリアドレスと等しいようなセグメントを設定する必要があります。このセグメントは386 ネイティブモードアプリケーションプログラムのロード前の初期設定時に設定されます。プログラムのセグメントはこのDOS-Extenderメモリ管理セグメントに続いて、ページテーブルにマッピングされます。したがってプログラムのセグメントのリニアベースアドレスは、マシンの全物理メモリ量を超えたリニアアドレスから始まるでしょう。DOS-Extenderはメモリの割当てとマッピングを１度に１物理ページずつ行うので、セグメント内の物理ページは、必ずしも連続する必要はありません。

7.1.2　ローカルディスクリプタテーブルセグメント

ローカルディスクリプタテーブル(LDT) には、アプリケーションプログラムの所有するセグメントが格納されています。プログラムの最初のロード時に設定されるLDT セグメントの一覧表を表3-1 に示します。DOS-Extenderは、INT21hファンクション 48h　システムコールでメモリ割当てが要求されるたびに、新しいセグメントをLDT 内に作成します。

7.1.3　グローバルディスクリプタテーブルセグメント

グローバルディスクリプタテーブル(GDT) には、DOS-Extenderの所有するセグメントが格納されます。これらのセグメントは通常、アプリケーションプログラムによって使用されることはありません。GDT 中に維持される一部のセグメントを表7-1 に示します。

表7-1　GDTセグメント

セグメント セレクタ値	説　明
0028h	LDT システムセグメント。このセグメントは、LDTRレジスタにロードされるセグメントセレクタ値である。
0030h	LDT をマッピングする書込み可能データセグメント。このセグメントは、DOS-ExtenderがLDT のセグメントディスクリプタントを変更するために使用する。
0038h	GDT をマッピングする書込み可能データセグメント。このセグメントは、DOS-ExtenderがGDT のセグメントディスクリプタを変更するために使用する。
0040h	セグメント内のオフセットが物理メモリアドレスに等しくなるようにメモリをマッピングする書込み可能データセグメント。このセグメントは、DOS-Extenderがシステムページテーブルの保持と変更のために使用する。
0050h	IDT(割込みディスクリプタテーブル) をマッピングする書込み可能データセグメント。このセグメントは、DOS-ExtenderがIDT の割込みディスクリプタを変更するため使用する。
0060h	MS-DOSで使用される下位１MBをマッピングする書き込み可能データセグメント。MS-DOS上での実行時、このセグメント内のオフセットは、物理メモリアドレスと等しい。

GDT(FMTOWNS用)

セグメントセレクタ値	説　明
0100h	SYSTEM ROM(code)
0108h	SYSTEM ROM(data)
0110h	TOWNS BIOS(code)
0118h	TOWNS BIOS(data)
0120h	TOWNS VRAM（16,32K色モード）
0128h	TOWNS VRAM（256色モード）
0130h	スプライト　パターンRAM
0138h	漢字ROM
0140h	波形RAM（PCM音色RAM）
0148h	OS ROM

7.2　メモリ割当て

DOS-Extenderは、メモリの割当てと解放を常に4KBページ単位で行います。アプリケーションプログラムは、LOW メモリ（1MB未満のメモリ）とHIGHメモリの両方を使用できます。

メモリをアプリケーションプログラムに割り当てる時、LOW メモリが割り当てられてから、次にHIGHメモリが割り当てられます。さらに、LOW メモリは、最初に割り当てられる時は必ず連続して割り当てられます。これは、リアルモードコードと386 ネイティブモードコードの両方を持つプログラムを作成する場合に意味があります（第8章参照）。もちろん、いったんプログラムが不要になったメモリを解放すれば、それ以後のメモリ割当てコールは、LOW メモリまたはHIGHメモリのどちらかを、特定の順序に従うことなく入手できます。

7.2.1　LOW メモリの使用

LOW メモリは、DOS-Extenderが初期設定時にDOS を呼び出すことによって入手するメモリブロックから割り当てられます。したがって、割当て可能なLOW メモリの量は、初期設定時に決定されます。デフォルトで、DOS-Extenderは使用可能なすべてのLOW メモリを入手します。

-MINREALコマンドラインスイッチと-MAXREALコマンドラインスイッチを、リンク時または実行時に使用して、DOS-Extenderに対してLOW メモリの一部を強制的に未使用のままにさせることができます。これは、プログラムが子プログラムに対して、EXECを実行したい場合に必要です。子プログラムをメモリにロードできるように、十分なLOW メモリが未使用である必要があります。

DOS-Extenderは自分自身が使うためのLOW メモリを約100 KB要求します。DOS-ExtenderのLOW メモリの使用は、-MINIBUF、-MAXIBUF、-NISTACK、-ISTSIZE、CALLBUFSコマンドラインスイッチに影響され、これらのスイッチでどれかがDOS-Extenderのデフォルト値より大きいLOW メモリデータバッファを割り当てるのに使用されると、DOS-Extenderが使用するLOW メモリ量は増加します。

第8章 リアルモードと386ネイティブモードコードの混在

8.1 概要

一つのプログラム中にリアルモードコードと386 ネイティブモードコードを両方使う場合には、解決しなければならない問題が三つあります。第一の問題は、リアルモードコードが必ずLOW メモリ内で終了するようにしながら、リアルモードと386 ネイティブモードの両コードをメモリに格納することです。第二の問題は、リアルモードと386 ネイティブモード間でデータを受け渡す方法です。最後の問題は、両モード間でプロシジャコールを行う方法です。これらの問題は、それぞれ、独立な複数の解決策を持っています。本章では、可能な選択肢について説明し、選択基準をいくつか示します。

8.2 プログラムのローディング

相互に通信の必要なリアルモードコードと386 ネイティブモードコードをロードするのに使用できる選択肢は三つあります。

第一の方法は、リアルモードコードと386 ネイティブモードコードをリンクして一つのタスクイメージを作成する方法です。この方法は概念的には最も簡単な方法ですが、コードが正しくロードされたことを確認するために、リンク時のセグメントの順序づけと、ロード時のメモリ割当てに関する構成作業が必要になります。さらにこの方法は、リアルモードコードがアセンブラで書かれている場合にだけうまく機能します。その理由は、同一リンク内でリアルモード用と386 ネイティブモード用のコンパイラの実行時ライブラリを結合しようとすると、結果として多くのシンボルが多重定義されます。しかし、通常、何らかのリアルモードコードを必要とする386 ネイティブモードプログラムは、既存のリアルモードハードウェアドライバか同等のプログラムと通信するために若干のリアルモードコードを必要とするだけです。この問題の解決法としてアセンブラ言語を使用します。

第二の方法は、まず386 ネイティブモードプログラムを起動して、次にそのプログラムにEXECシステムコールさせて別のリアルモードプログラムをメモリにロードする方法です。386 ネイティブモードコードとリアルモードコードは、全く別の実行ファイルなので、別々のコンパイラを使うか、別々の高級言語を使用しても問題はありません。この方法で発生する構成上の問題はただ一つで、リアルモードプログラムのロードに必要なLOW メモリを十分残すことです。

この方法の難点は、二つのプログラム間のコミュニケーションを確立することです。これは、プログラム同士はリンクされなかったので、互いに相手のプログラム中のプロシジャとデータのアドレスを持たないからです。

第三の方法は、リアルモードプログラムを起動しそのプログラムが386 ネイティブモードに対するEXECを実行する方法です。この場合、リアルモードプログラムが最初にロードされるので、LOW メモリの割当てについての問題は発生しません。しかし、この方法でも第二の方法と同じコミュニケーションの問題があります。さらにこの方法には、デバッガかDOS-Extenderを実行するために、386 ネイティブモードプログラムをデバッグしたいのかによってEXECシステムコールを変更しなければならないというやっかいな問題もあります。

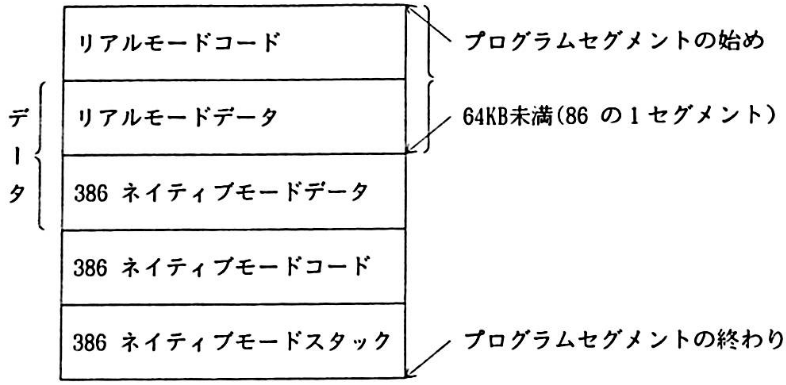
8.2.1 リアルモードコードと386 ネイティブモードコードのリンク

同一プログラム内にリアルモードコードと386 ネイティブモードコードをリンクする方法は、リアルモードコード量が比較的少なく、しかもアセンブラ言語で作成できる場合に適しています。リアルモードコードはUSE16 セグメント内に置かれ、 80386をターゲットとしてアセンブルされます。このように作成したコードはリアルモードで正しく実行されます。80386 でのリアルモードは386 ネイティブモードと同じアドレッシングモードとオペランドサイズを使用でき、アセンブラは必要なオーバライドバイト等を自動的に作成するので、アドレッシングモード、オペランドサイズ、または使用可能な命令に対する制限は何もありません。プログラムはこの後、386 ネイティブモードプログラムの場合と同様に、80386 をターゲットマシンとしてリンクされます。

しかし、必ず守るべき制限事項がいくつかあります。386 ネイティブモードプログラムを作成する場合、386 ネイティブモードプログラムを生成する 386 | LINKは、プログラム内のすべてのコードセグメント、データセグメントおよびスタックセグメントを結合して、単一の大きなプログラムセグメントを作成します。セグメント化されたリンクはサポートされないので、プログラムによって必要とされるセグメント操作は存在しません。これは、リアルモードコードはすべて単一セグメント中に置く必要があることと、NEARコールとNEARジャンプだけがリアルモードコードによって実行できることを意味します。

次に解決する問題は、すべてのリアルモードのコードとデータはLOW メモリにロードされることを確認しなければならないことです。言うまでもなく、リアルモードコードはリアルモードで実行するために、LOW メモリにロードしなければなりません。さらにプログラムのリアルモード部分と386 ネイティブモード部分との間の通信に使われるプログラムのデータの部分集合があります。このデータは、リアルモードコードがこのデータにアクセスできるように、LOW メモリにロードする必要があります。

LOW メモリにロードするコードとデータはすべて、プログラムのリンク時に.EXPファイルの先頭に置かなければなりません。これは 386 | LINKのセグメント順序規則に従って行われます（386 | LINK使用手引書を参照してください）。リアルモードコードと、リアルモードでアクセスすることが必要なデータは、それら自身のセグメントに格納する必要があります。リンカに処理される最初のオブジェクトファイルのセグメントの順序が、セグメントクラスおよび他のリンカ順序規則と共に、.EXPファイルのセグメントの順序を決定します。最終的に、以下のように順序付けされた.EXPファイルになります。



リンカはリンク時にすべてを結合してひとつの大きなセグメントを作成するので、オフセットはすべてこの大きなプログラムセグメントの開始からのオフセットとなります。リアルモードの実行時、CSセグメントとDSセグメントは、プログラムセグメントの先頭のリアルモードセグメントパラグラフアドレスに設定され、リアルモードコードとデータの中のアドレスの通常の参照は正しく機能します。要求される事項はただ一つで、リアルモードコードとデータを加えたサイズが64KB未満でなければならないことです。

リアルモードプロシジャが制御権を獲得すると、CSレジスタはプログラムセグメントの先頭のリアルモードパラグラフアドレスに設定されます。リアルモードプロシジャはこのアドレスをCSからDSにコピーすることが必要なだけで、プログラム内のすべてのリアルモードデータに直接アクセスできます。

386 ネイティブモードでの実行時、CSとDSの中のセグメントセレクタ値によってアクセスされるセグメントディスクリプタは、通常プログラムセグメント全体をマッピングします。したがって両モードのデータセグメント内のデータを直接参照できます。

データ参照を正しく機能させるために、アセンブラのASSUME命令を使用するのはあまり良い方法ではありません。その理由は、プログラムのソースコードはいくつかのセグメントに分割されますが、それらはリンク時単一のセグメントに結合されるからです。リンカは、単一プログラムセグメントの先頭アドレスを基準にしてすべてのアドレスを再配置するので、DSセグメントレジスタが上述のように設定されている時は、すべての参照はリアルモードと386 ネイティブモードの両モードで正しく機能します。この場合の問題はただ一つ、アセンブラが何か言ってこないようにこの事実をアセンブラに通知することです。この問題の一つの解決策は、ASSUME命令を使い、DSレジスタが現在アクセスしたいデータセグメントを指していることをアセンブラに通知することです。しかしもっと良い方法は、リアルモードデータセグメントと386 ネイティブモードデータセグメントの両方を1グループ内に置き、次にASSUME命令を使いアセンブラに対してDSレジスタがこのグループを指すことを知らせる方法です。この方法によりアセンブラは、ASSUME命令をプログラム中に指定しないで、リアルモードと386 ネイティブモードの両方のデータ参照を正しくアセンブルすることができます。これが386 ネイティブモードコードセグメントを386 ネイティブモードデータセグメントの後ではなくリアルモードデータセグメントの次に置く理由です。

セグメントはリンカのセグメント順序付け規則に従って１グループにして格納されます。
いったんセグメント順序付けが正しく行われると、残る問題はすべてのリアルモードのコードとデータが必ずLOW メモリにロードされることを保証することです。これはプログラムのリンク時か、またはプログラムがDOS-Extender上で実行する時、-REALBREAKコマンドスイッチを使用することによって実現されます。このコマンドスイッチは、連続したLOWメモリにロードする必要がある.EXPファイルのサイズを指定します。一般に、プログラムリンク時にこのスイッチを使用することが最も簡単で、リアルモードデータセグメントの最後にあるパブリックシンボル名を与えることによってシンボリックに処理できます。このスイッチがDOS-Extender上でプログラムの実行時に指定される場合、絶対数の指定の必要があります。これは、リンカで作成されたマップファイルを使用してリアルモードのコードとデータのサイズを算出することが必要であることを意味します。

8.2.2　386 ネイティブモードプログラムからリアルモードプログラムの実行（EXEC）

第２の方法は、EXECシステムコールを使って386 ネイティブモードプログラムにリアルモードプログラムをメモリにロードさせる方法です。リアルモードプログラムが制御を受け取ると、386 ネイティブモードプログラムの呼出しはメモリにロードされているので可能です。この方法が使用されるとリアルモードプログラムのサイズ制限やそのプログラムの持つセグメント数の制限はありません。またこのアプローチには、特殊なセグメント順序付けの問題もありません。必要なことはただ一つ、リアルモードプログラムのロードに十分なLOW メモリを必ず確保するように、-MINREALスイッチや-MAXREALスイッチを用いて、386 ネイティブモードプログラムをリンクまたは実行することです。

DOS-Extenderシステムコール:25C3hは、システムファンクション:4Bhの代わりに、リアルモードプログラムをEXECするのに使用されます。これは、システムファンクション:4Bhは既存のプログラムとの互換性を保持するために、EXECの実行中は標準MS-DOSの割込みベクタの設定（6.6.1節を参照のこと）にシステムハードウェア割込みを返すためです。これは、EXECで実行されたプログラムがメモリ中にまだ常駐している時に、再度386 ネイティブモードプログラムが制御を得ようとする時に不適当です。386 ネイティブモードプログラムが実行中は、ハードウェア割込みのベクタはDOS-Extenderが予定している場所になければいけません。システムコール:25C3hはハードウェア割込みの再配置をせずにEXECを実行するため、リアルモードと386 ネイティブモードの混在したプログラムで正しいコールをするために使用されます。

この方法によれば、設定とプログラムロードはずっと簡単になりますが、プログラム間の通信はより難しくなります。その理由は両プログラムはお互い相手のプロシジャとデータのアドレスを保持せず、さらにリアルモードプログラムは386 ネイティブモードプログラムのデータには直接アクセスできない（そのデータはLOW メモリにないかもしれない）からです。これらの問題の解決方法は、第8.3 節と第8.4 節で述べます。

8.2.3 リアルモードプログラムから 386ネイティブモードプログラムの実行(EXEC)

第三のプログラムロード方法は、MS-DOSシステムファンクション4Bh:を使用して386 ネイティブモードプログラムに対してEXECを実行する標準リアルモードプログラムを起動することです。いったん386 ネイティブモードプログラムが制御権を獲得すると、リアルモードプログラムの呼出しはメモリにロードされているので可能です。この方法ではプログラムのロード時、特別なことは何も必要がありません。メモリの割当てやプログラム配置の問題がありません。しかし、第二の方法と同様に、プログラム間のコミュニケーションが困難です。詳細は、以下の節を参照してください。

8.2.4 二つのＰＳＰの管理

リアルモードや386 ネイティブモードプログラムと違って混在モードのプログラムの複雑さは、各モードでPSP(Program Segment Prefix) を持つことです。プログラム情報を管理するためにMS-DOSで使用されるPSP については、3.3 節および参考文献３で説明しています。

MS-DOSは"カレントPSP "と言う概念を持ちます。プログラムが実行のためにメモリにロードされるとそのPSP はカレントPSP にセットされます。そのプログラムが次のプログラムを実行(EXEC)させる時、子プログラムのPSP がカレントPSP として設定されます。再び親プログラムのPSP が設定されるのは、子プログラムから制御が復帰した時です。

リアルモードと386 ネイティブモードのプログラムが、お互いに呼び出し合うことを目的としてメモリ中に別々にロードされる時、（EXECコールを使い）２番目にロードされたプログラムのPSP がカレントPSP として使われます。これはプログラムがメモリにロードされている間、実際に実行しているいないにかかわらず使われます。通常このことは問題ないように見えますが、望ましくない副作用を持つ振る舞いを起こします。PSP の機能の一つはプログラムでオープンされたディスクファイルのリストを管理することです。プログラムが終了した時、オープンされたすべてのファイルは自動的にクローズされます。このことが問題を引き起こす場合について、次の例を考えながら見てみましょう。386 ネイティブモードプログラムを最初に実行し、通信するリアルモードプログラムを起動したと仮定します。もしリアルモードプログラムの実行中にファイルが（どちらかのプログラムで）オープンされているとそれらはリアルモードプログラムのPSP で参照され、386 ネイティブモードのPSP ではありません。リアルモードプログラムが終了し、すべてのファイルが自動的にクローズされると、386 ネイティブモードプログラムはそのファイルをアクセスすることができません。

もし上記の例やPSP のその他の使用で問題を引き起こす場合、各プログラムは制御を受け取った時に自分自身のPSP に変更するという対応をしなくてはなりません。これはシステムコールのGet PSP とSet PSP で行うことができます。Set PSP ファンクション(:50h)はリアルモードでも386 ネイティブモードでも同じであり、カレントPSP のアドレスをセットするためにPSP のリアルモードパラグラフアドレスをMS-DOSへ渡します。Get PSP ファンクション(62h) は両モードで処理が異なります。リアルモードでは、PSP のリアルモードパラグラフアドレスを返し、386 ネイティブモードではプログラムPSP の386 ネイティブモードセグメントセレクタ(0024h) を返します。PSP のリアルモードアドレス（Set

PSP ファンクションに必要）を入手する方法は、PSP セグメントのベースアドレスを入手するためにシステムコール：2508h を使います。PSP 切換えのプログラム例は付録－Cで示すグラフィックライブラリを使用した混合モードのプログラムにあります。

8.3 モード間のデータ受け渡し

DOS-Extenderは、モード間プロシジャコール時のデータ受け渡し用の方法を三つ用意しています。第一の方法は、データをスタック上で受渡すだけです。モード間プロシジャコールがリアルモードから386 ネイティブモードに出される時、同じスタックが保持されます。このため、リアルモードコードによってスタックに入れられたデータは、386 ネイティブモードプロシジャによって自動的にアクセス可能となります。モード間プロシジャコールが386 ネイティブモードからリアルモードに変更されると、DOS-Extenderに与えられる呼び出しパラメータの一つは、386 ネイティブモードプロシジャのスタックからリアルモードプロシジャに与えらたスタックへコピーするデータの量です。モード間プロシジャコール時のスタックの設定と使用法については、第8.4 節でさらに詳しく説明します。

DOS-Extenderは、初期設定時に、LOW メモリに常駐していることが保証されているバッファをオプションで割り当てます。このバッファサイズは０から64KBの範囲であり、-CALLBUFS コマンドラインスイッチにより制御されます。このバッファはアプリケーションが自由に使用できる専用バッファであり、DOS-Extenderがこのバッファを使用することはありません。実行時にこのバッファのリアルモードアドレスと386 ネイティブモードアドレスの両方を入手するためにシステムコールが用意されています。したがって、モード間プロシジャコールはこのバッファ内でデータの受け渡しをするのに必要なプロトコルを定義することができます。モード間でのデータ受け渡しの第三の方法は、80386 レジスタを使用することです。DOS-Extenderは第8.4 節で述べた場合を除き、モード間プロシジャコール時にはレジスタを変更しません。

8.4 モード間制御転送

リアルモードと386 ネイティブモード間の制御の転送方法には二つあります。第一の方法はDOS-Extenderのモード間プロシジャコールを使用する方法です。第二の方法はソフトウェア割込みを使ってモードを切換える方法です。モード切換え時のスタック設定方法はどちらも同じ方法です。

リアルモードから386 ネイティブモードへの切換え時、386 ネイティブモードプロシジャはリアルモードプロシジャの使用していたのと同じ物理スタックを入手します。この入手には、SSをGDT（ セグメント:0060h）内のMS-DOSの最初の１MBをマッピングするDOS-Extenderセグメントに示すように設定する必要があります。その後、ESP がリアルモードスタックのリニアアドレスに設定されます。

386 ネイティブモードからリアルモードへの切換え時、リアルモードプロシジャは、DOS-Extenderによって内部的に維持されているスタックメモリバッファを与えられます。これらの内部スタックバッファのサイズと数は、-NISTACKコマンドラインスイッチと

-ISTKSIZE コマンドラインスイッチによって制御されます。デフォルトでは、1KB構成のバッファが4個指定されます。386 ネイティブモードからリアルモードへの切換えを、入れ子にして行うたびに、別々のバッファが必要です。さらに、ソフトウェア割込みまたはモード間プロシジャコールによって起動されたモード切換え時を含めて、いつでも発生しうるハードウェア割込みで使える余分のバッファを一つ用意しておく必要があります。したがって、デフォルトの4個のバッファは、モード切換えを3レベルより深く入れ子にするプログラム以外のすべてのプログラムに対して十分な数です。この計算時、386 ネイティブモードで発生するソフトウェア割込み（DOS システムコールとBIOSシステムコールを含む）は、すべてリアルモードへの切換えをすることを心にとめておいてください。

8.4.1 モード間プロシジャコール

DOS-Extenderシステムファンクション：250Eh と2510h は、386 ネイティブモードからリアルモード関数を呼び出すために用意されています。二つのシステムコールの違いは、ファンクション2510h はセグメントレジスタを含むすべてのレジスタの値をコール側で指定することができ、ファンクション250Eh はDSレジスタの内容をCSレジスタと同じ値にし、その他のセグメントレジスタの値は不定となることです。このファンクションは入力として、コールするプロシジャのFAR リアルモードアドレス（セグメント：オフセット）と、386 ネイティブモードスタックからリアルモードプロシジャに与えられるスタックへコピーする4バイト構成の DWORD数のカウントを必要とします。

リアルモードプロシジャが制御権を獲得すると、(1)すべての汎用レジスタの値は変更されず、(2)CS:IP はプロシジャのエントリポイントを示し、(3)SS:SP はDOS-Extenderの内部スタックバッファのアドレスに設定され、(4)DSレジスタはCSと同じ内容に初期設定され（リアルモードと386 ネイティブモードのコードが一緒にリンクされたプログラムにとっては望ましいため）、そして(5)ES、FSとGSレジスタの内容は壊されます（ファンクション:2510 h用には、コール側はリアルモードのDS/ES/FS/GS セグメントレジスタの値を設定しなくてはなりません）。リアルモードスタックサイズは、-ISTKSIZE コマンドラインスイッチによって制御され、デフォルト値は1KBです。スタックは、386 ネイティブモードスタックからコピーするよう指定されたデータと、その後にFAR リアルモード復帰アドレスを持ちます。リアルモードプロシジャはその処理を終了すると、FAR リターンを実行して戻ります。制御権は386 ネイティブモードプロシジャに戻された時、汎用レジスタの内容はすべてリアルモードプロシジャの残した値のままで、セグメントレジスタの内容はすべて元の386 ネイティブモード値に戻されます。プロセッサフラグの内容は、システムコールの成否を識別するための386 ネイティブモードに返されるキャリーフラグを除き、両方向で保持されます。システムコール：250Ehと2510hの指定方法は付録－Eを参照してください。

リアルモードコードが386 ネイティブモードプロシジャをコールしたい場合、実行時に動的にアドレスが決まるDOS-Extenderのリアルモードルーチンに対してFAR コールを実行します。このDOS-Extenderルーチンのアドレスはシステムファンクション：250Dh を386 ネイティブモードからコールすることによって得られます。このルーチンに渡される引数は、386 ネイティブモードプロシジャのFAR アドレス（DWORD オフセットとWORDセグメントセレクタ値）と、その後に続く386 ネイティブモードプロシジャに渡されるパラメータで構成されます。

このコールが実行される時、DOS-Extenderはリアルモード復帰アドレスと386 ネイティブモードプロシジャアドレスをスタックからポップして保存し、その後386 ネイティブモードに切換え、386 ネイティブモードプロシジャに制御を渡します。386 ネイティブモードプロシジャが制御を獲得すると、(1)すべての汎用レジスタの内容は保持され、(2)SS:ESPはリアルモードスタックのアドレスにセット（これは GDT内のMS-DOSメモリの最初の1MBをマッピングするDOS-ExtenderセグメントにSSを設定し、ESP をスタックのリニアアドレスに設定することによって行われます）し、(3)DS／ES／FS／GSは、386 ネイティブモードプログラムが起動した時の値に設定されます。したがって、パラメータはスタックにのせて386 ネイティブモードプロシジャに直接渡すことができます。

386 ネイティブモードプロシジャはその処理を終了すると、FAR リターンを実行して終了します。DOS-Extenderは制御権をリアルモードプロシジャに戻します。この時、汎用レジスタの内容はすべて386 ネイティブモードプロシジャの残した値のままで、セグメントレジスタのすべての内容は元のリアルモードの値に戻されます。プロセッサフラグの内容はコールの、どちらの方向でも保存されます。詳細は付録－Eを参照してください。

8.4.2 割込みを使ったモードの切換え

ソフトウェア割込みでモードを切換えたいプログラムは、モード切換え用の割込みベクタを継承する初期設定コードを持っていなければなりません。二つの割込みベクタが最小限必要です。一つは、386 ネイティブモードからリアルモードに切換えるのに使用される割込みであり、“リアル割込み”と呼びます。もう一つは、リアルモードから386 ネイティブモードに切換えるのに使用される割込みであり、“386 ネイティブ割込み”と呼びます。 初期化ルーチンは386 ネイティブモードで実行されなくてはなりません。これは、割込み発生時にいつでもある386 ネイティブモードハンドラに制御が渡るように設定します。システムファンクション2506h によって386 ネイティブ割込みを引きつぎます。リアル割込みはある割込み発生時に、あるリアルモードハンドラに制御が渡るように設定するシステムコール：2506h によって引きつがれます。

別の手段として、リアルモード初期化ルーチンはMS-DOSファンクション：25h をコールしてリアル割込みを引きつぐことができます。リアルモードのコードは、386 ネイティブの割込みを引きつぐことはできません。

いったんこの初期設定が実行されると、386 ネイティブモードコードはリアル割込みをソフトウェア割込みとして発行することによって、いつでも制御権をリアルモード割込みハンドラに移すことができます。リアルモード割込みハンドラが制御権を獲得すると、すべての汎用レジスタとプロセッサフラグの内容は、386 ネイティブモードで割込みが発行されたときと同じ値を持ちます。ただし、割込みフラグはオフ（割込み禁止）になります。SS:SP レジスタの内容は、DOS-Extenderによって内部的に維持されているスタックメモリバッファを指すように設定されます。このバッファのサイズは、-ISTKSIZE コマンドラインスイッチによって制御されます。デフォルト値は1KBです。DS／ES／FS／GSの各レジスタの内容は破壊されます。

処理を終了すると、リアルモードハンドラはIRET命令を実行して終了します。

DOS-Extenderは制御権を386 ネイティブモードプロシジャに戻します。この時、すべての汎用レジスタとプロセッサフラグ（割込みフラグとトラップフラグを除く）の内容は、リアルモードハンドラの残した値のままです。全セグメントレジスタの元の386 ネイティブモード値が復元されます。

リアルモードコードは386 ネイティブ割込みをソフトウェア割込みとして発行することによって、いつでも制御権を386 ネイティブモード割込みハンドラに移すことができます。386 ネイティブモードハンドラが制御権を獲得する時、すべての汎用レジスタとプロセッサフラグの内容は、リアルモードで割込みが出された時と同じ値を持ちます。ただし、割込みフラグはオフ（割込み禁止）となります。SS:ESPレジスタの内容は、リアルモードスタックを指すように設定されます（この設定は、GDT 内のMS-DOSメモリの最初の１MBをマッピングするDOS-ExtenderセグメントにSSを設定し、ESP をスタックのリニアアドレスに設定することによって行います）。これによってスタックを経由してデータの受け渡しができます。DS／ES／FS／GSレジスタの内容は386 ネイティブモードプログラムが起動した時の値に設定されます。386 ネイティブモードハンドラが制御を受け取ると、スタックは次のように設定されます。

00h	DWORD	386 ネイティブモードのEIP リターン値
04h	DWORD	386 ネイティブモードのCSリターン値
08h	DWORD	386 ネイティブモードのEFLAGSリターン値
0Ch	WORD	リアルモードIPリターン値
0Eh	WORD	リアルモードCSリターン値
10h	WORD	リアルモードFLAGS リターン値
12h		リアルモード割込み発行前に置かれたいくつかのパラメータ

386 ネイティブモードハンドラは処理を終了すると、IRETD 命令を実行し終了します。そのあとDOS-Extenderは制御権をリアルモードプロシジャに戻します。この時、すべての汎用レジスタとプロセッサフラグ（割込みフラグとトラップフラグを除く）の内容は386 ネイティブモードハンドラの残した値のままです。全セグメントレジスタのリアルモードの値が復元されます。

8.5 典型的なプログラム編成

リアルモードと386 ネイティブモードコードを使用するプログラムを設計する場合、希望に応じて上記の方法を混ぜたり、併用したりすることが可能です。本節では、混在モードのプログラムを編成するのに簡単で有用な方法をいくつか説明します。

単一の.EXPファイルの中でリアルモードと386 ネイティブモードコードを混用するプログラムは、典型的に次のような初期設定ルーチンを持っています。(1)システムコール：250Fh でプログラムセグメントの先頭のMS-DOSアドレスを入手し、(2)システムファンクション：250Dをコールして、リアルモードから386 ネイティブモードへの切換えに用いるDOS-Extenderルーチンのリアルモードアドレスを入手し、この情報をリアルモードコードでア

クセスできるグローバルデータに保存します。これによってプログラムはDOS-Extenderモード間プロシジャコールでモードの切換えに必要なすべての情報を持つことができます。データ転送は、スタックまたはリアルモードコードでアクセスできるグローバルデータのいずれかを使用して行われます。

386 ネイティブモードプログラムが25C3h EXECシステムコールを使ってメモリにリアルモードプログラムをロードするように設定されたプログラムは、典型的に、以下のステップを実行する初期設定ルーチンを持ちます。

1. システムファンクション：250Dh をコールして、リアルモードから386 ネイティブモードへの切換えに使用されるDOS-Extenderルーチンのリアルモードアドレスを入手し、モード間コール時のデータ受け渡しに使用されるデータバッファのリアルモードと386 ネイティブモードアドレスを入手する。
2. リアルモードプログラムを実行するためにシステムファンクション:25C3hを呼び出し、コマンドラインを使い、(a)386 ネイティブモード呼出しに利用されるDOS-Extenderルーチンのリアルモードアドレス、(b)内部データバッファのリアルモードアドレス、(c)386 ネイティブモードプログラム内のインターフェイスルーチンの386 ネイティブモードアドレスを受け渡す。
3. リアルモードプログラムがその後制御を受け取り、それ自身の初期化を実行し、386 ネイティブモードへ制御を戻すために386 ネイティブモードのインタフェースルーチンをコールし、リアルモードプログラムのインターフェイスルーチンのリアルモードアドレスを受け渡す。

二つのプログラムは継続して実行することができ、相互の呼出しや、スタックまたは内部データバッファでデータの受け渡しもできます。プログラムが処理を終了した時、386 ネイティブモードプログラムはステップ３で行われたインタフェースルーチンのコールから復帰します。リアルモードプログラムはMS-DOSファンクション：4Ch により終了し、386 ネイティブモードルーチンはEXECシステムコール25C3h に続く命令から再び実行を再開します。

最初にリアルモードプログラムを実行させ、次に386 ネイティブモードプログラムを実行するプログラムは、典型的に次の初期化手続きを実行します。

1. 一つ以上の割込みが、標準的なリアルモードMS-DOS割込みベクタ獲得設定システムファンクションをコールすることによって継承される。これらの割込みは、制御権をリアルモードプログラムに戻すために386 ネイティブモードプログラムによって使用される。
2. 標準的なMS-DOSシステムEXECファンクション:4Bhが、386 ネイティブモードプログラムのロードと起動を行うためにコールされる。
3. 386 ネイティブモードプログラムは、システムファンクション：2506h を使って一つ以上の割込みを継承する。これらの割込みは、制御権を386 ネイティブモードプログラムに戻すためにリアルモードプログラムによって使用される。

いったんこの初期設定が完了すると、リアルモードと386 ネイティブモードコードは、モードを切換えるのに前もって準備しておいたソフトウェア割込みを使うだけです。データはスタックまたはレジスタによって受け渡されます。386 ネイティブモードコードはモード間コールデータバッファのアドレスを入手するために、システムファンクション：250Dh を使用できます。その後386 ネイティブモードコードは、データバッファがソフトウェア割込みを使ったモードの切換え時に、データを受け渡すために使用できるようにリアルモードコードにアドレスを渡します。

8.6　独自のリアルモードシステムコールの作成

DOS-ExtenderはマニュアルにMS-DOSシステムコールのすべてとほとんどのBIOSシステムコールを拡張しており、このため386 ネイティブモードからそれらを直接利用することができます。しかし、プログラムによってはマニュアルに記載されていないMS-DOSファンクションにアクセスすることを必要としたり、システムブート時にメモリにインストールされるネットワークドライバといった別のプログラムへのシステムコールをつくる能力が必要となります。

入出力用に汎用レジスタ（セグメントレジスタは除く）のみを使用するシステムコールは、リアルモードからと同じように386 ネイティブモードから利用できます(6.1節を参照)。もしシステムコールがセグメントレジスタを使用するなら、386 ネイティブモードプログラムに追加処理が要求されます。

DOS-Extenderのシステムコール:2511hは、セグメントレジスタを含むすべてのレジスタを使ってリアルモード割込みを発生させるのに使われます。また、リアルモード割込みハンドラが残したセグメントレジスタの値を返します。このシステムコールは、リアルモードで発行できるすべての割込みの発行を可能にします。

残りの問題はデータのバッファリングです。セグメントレジスタを使用するシステムコールは、データを含むバッファへのポインタを必要とするため、通常データのバッファリングをします。リアルモードコードがそのアドレスを示せるようにするためこのバッファはLOW メモリにある必要があります。LOW メモリにあることが保証されるデータバッファは、LOW メモリに386 ネイティブモードプログラムの一部分を強制的にロードするか(8.2.1）、またはメモリを直接割り当てるためにMS-DOSをコールして(8.7節）割り当てられます。しかしおそらく最も単純な方法は、-CALLBUFS コマンドラインスイッチを使って初期化時にLOW メモリ内にデータバッファを割り当て、実行時にシステムファンクション:250Dhを使ってデータバッファのリアルモードと386 ネイティブモードのアドレスを獲得することです。

システムコールがデータバッファへのポインタを返す時、先頭の１MBのLOW メモリをマッピングするセグメントセレクタ:0034hを使ってデータはアクセスできます。

8.7　LOWメモリの割当て

MS-DOSの標準メモリ割当てのシステムコール（48h, 49h, 4Ah ）はDOS-Extenderですべて操作されます。これらのシステムコールはHIGHメモリまたは LOWメモリの両方のメモリブロックを割り当てたり解放できます。DOS-Extenderによって割り当てられるメモリブロックは４KBメモリページで構成され、物理的に連続してもしなくてもかまいません（第７章を参照）。

プログラムによっては LOWメモリ内にあるか、または物理的に連続していることが保証されるメモリブロックを要求します。DOS-Extenderはシステムコール(25C0h, 25C1h, 25C2h) によりMS-DOSメモリ割当て機能を呼び出す機能を提供しています。これらのシステムコールで割り当てられたメモリは、先頭１MBの LOWメモリを示すセグメントセレクタ：0034h を使い386 ネイティブモードからもアクセス可能です。

初期化時にDOS-Extenderに LOWメモリを残させるためには、-MINREAL/-MAXREAL コマンドラインスイッチでその値を指定する必要があります。この指定がないと、 LOWメモリ割当てができなくなり、"out of memory error" が返されます。

第9章 互換性

9.1 HIGHメモリの割当て

DOS-Extenderは386 ネイティブモードアプリケーションプログラムに対し他のプログラムが使用しないHIGHメモリを割り当てることができます。これらのメモリ割当て方式の詳細は第7章を参照してください。

HIGHメモリを使用しても、HIGHメモリの割当てに標準の方法を用いないプログラムがシステムにインストールされている場合は、矛盾を避けるためDOS-Extenderの使用するHIGHメモリ量を制限する必要があります。この制限は、HIGHメモリ内のどのアドレスが他のプログラムによって使用されているかを調べ、次に-EXTLOW と-EXTHIGHのコマンドラインスイッチを使ってDOS-Extenderの使うHIGHメモリを制限することによって行われます。

9.2 アドレスライン20の有効化

ハードウェアのアドレスライン20を有効化または無効化する機能を持ちます。アドレスライン20が無効にされると、80386 によってバスに載せられる物理アドレスは、外部ハードウェアによって20ビットまでに切り捨てられます。この切り捨ては1MBのアドレスラップラウンドに依存するMS-DOSプログラムと互換を維持するためです。したがって、アドレスライン20はMS-DOSのリアルモード実行中は無効となります。386 ネイティブモードプログラムを正しく動作させるにはアドレスライン20は、1MBを超えるメモリをアクセスできるようにするため、有効にしておく必要があります。省略時、DOS-Extenderは、初期設定時にアドレスライン20を有効にし、アプリケーションプログラム実行中は有効のままにしておきます。プログラムが終了すると、DOS-Extenderはアドレスライン20を最初の設定（有効または無効）に戻します。

-A20コマンドラインスイッチは、DOS-Extenderに対して386 ネイティブモードの切換時にアドレスライン20を有効にし、リアルモードの切換時には無効になるように強制します。このスイッチを使う理由は二つあります。

第一は、1MBのアドレスラップラウンドに依存するリアルモードプログラムを386 ネイティブモードプログラムの実行中に動作可能にするためです（つまり、終了後も常駐するプログラムが386 ネイティブモードプログラムか実行するリアルモードプログラムのどちらかであるため）。ただし、このようなことはあまりありません。

第二の理由は、DOS-Extenderアプリケーションが実行中に-A20を有効または無効にできるメモリ常駐プログラムと互換を維持するためです。HIGHメモリアクセスプログラム、たとえばRAM ディスクドライバまたはディスクキャッシュドライバなどがこれを行う必要があるかも知れません。これらのプログラムの大多数は、BIOSのブロック移動関数コールを使って拡張メモリにアクセスします。 DOS-Extender はこれらのBIOSコールを横取りし、コール終了後にA20 を再び有効にしますので、これは問題になりません。ただし、これらのプログラムの一部は標準以外の方法でHIGHメモリにアクセスするので、そのようなプログラムによってA20が無効にされたことをDOS-Extenderは検出できません。このメモリ常駐プログラムがシステムにインストールされている場合には、DOS-Extenderの実行時、-A20スイッチを使用する必要があります。

付録A 386 | DOS-Extender コマンドライン

```
RUN386 switches name  param(s)
```

switches	386 | DOS-Extenderのデフォルトの動作を変更するためのオプション指定のコマンドラインスイッチ
name	実行する.EXPファイルまたは .REX ファイルの名前
param(s)	コマンドラインパラメータがネイティブモードアプリケーションプログラムに渡されます。

付録B エラーメッセージ

DOS-Extenderで発生の可能性のあるエラーを、その原因と処置と共に以下に示します。エラーメッセージはアルファベット順に説明します。

エラー：? Can't open executable file：filename

原因：　存在しない.EXP、.EXEまたは.REXのファイルを実行しようとした。

処置：　1).EXPファイルの名前の綴りを直す。

　　　　2).EXPファイルのある正しいディレクトリ名を指定する。

　　　　3)386｜LINKで.EXPファイルを作成する。

エラー：? Error reading file：filename

原因：　.EXPファイルが壊れている

処置：　リンクエラーのないことを確認して、.EXPファイルを再構築する。

エラー：? Insufficient memory to load program

原因：　使用可能なメモリよりプログラムの方が大きい

処置：　1)プログラムを縮小する。

　　　　2)メモリを追加する。

　　　　3)メモリ常駐プログラムをシステムから削除する。

エラー：? Load error

原因：　MS-DOSがリアルモード.EXEファイルをロードしようとした時に原因不明のエラーが発生した。

処置：　システムを再ブートする。

エラー：? Missing file name

原因：　実行する386 ネイティブモードプログラムを指定せずに、DOS-Extenderを実行しようとした。

処置：　実行するプログラムを収めた.EXPファイルの名前を指定する。

エラー：? Not an executable file : filename

原因：　ファイルが.EXP、.EXEまたは.REXの形式のファイルではない。

付録C　80386 プログラム例

386｜ASM/LINKの提供ディスクには、386 ネイティブモードMS-DOSアプリケーションプログラムのいくつかの例が格納されています。各プログラムは、(1)ファイル名拡張子“.ASM”を持つアセンブリ語ソースコードファイル、(2)ファイル名拡張子“.EXP”を持つ 386 ネイティブモードタスクイメージおよび(3)ファイル名拡張子“.BAT”を持ちプログラムのアセンブルとリンクに使用可能なMS-DOSバッチファイルの三つのファイルとして提供されます。

これらのプログラム例によって、システムコールを行う 386ネイティブモードアプリケーションプログラムの基本的な構造を具体的に理解できます。各々のプログラム例について以下に簡単に説明します。

RMHELLO　　80386 命令と32ビット命令オペランドを使用する簡単なリアルモードプログラム。

HELLO　　文字列“Hello world”を画面に表示する簡単なアプリケーション。このプログラム例はMS-DOSシステムコールの方法を示します。

ECHO　　コマンドライン引数を画面にエコーするプログラム。このプログラム例は 386 ネイティブモードプログラムからMS-DOSコマンドラインにアクセスする方法を示します。たとえば、以下のコマンド

```
RUN386 ECHO TESTING 1 2 3
```

は、画面に“TESTING 1 2 3”を表示します。

SCRTEST　　このプログラムは、セグメントセレクタ001Ch を使用して 386ネイティブモードプログラムから物理画面メモリへ直接アクセスする方法を示します。このプログラムを実行すると、画面がクリアされ、PC文字セットが画面上のメッセージボックス内に表示されます。

《このプログラム例は、そのままではアセンブルして動作可能なものではありません》

DOSDEMO　　このプログラムは 386ネイティブモードから種々のMS-DOSシステムコール（画面、キーボードとファイル入出力）を実行します。
《このプログラム例は、そのままではアセンブルして動作可能なものではありません》

TAIL　　本文の8.5.1 項で説明した、Microsoft マウス使用のリアルモードとネイティブモード混用プログラム。
《このプログラム例は、そのままではアセンブルして動作可能なものではありません》

付録D MS-DOSシステムコール

MS-DOSシステムコールは、 20hから 3Fhまでのソフトウェア割込みによって行われます。パラメータはレジスタによって受け渡され、結果を返すレジスタ以外はすべてコールを通じて保持されます。本付録では、使用できるすべてのMS-DOSシステムコールを説明します。割込みによっては二つ以上のファンクションを実行する割込みもいくつかあり、入力の一つ（通常、AHレジスタの値）は実行したいファンクションのファンクションコードです。マニュアルに記載されていない MS-DOS 割込みと多重ファンクション割込み用ファンクションが DOSに渡される時、すべての汎用レジスタの値は変わりませんが、DSとESのセグメントレジスタの内容は破壊されます（本文の第6.1 節参照)。

表　MS-DOSシステムコール一覧

割込み番号	ﾌｧﾝｸｼｮﾝ番号	ファンクション概要	ﾍﾟｰｼﾞ
20h	なし	プログラム終了	55
21h	00h	プログラム終了	55
21h	01h	文字入力とエコー　【注】	55
21h	02h	文字出力　【注】	56
21h	03h	補助入力	56
21h	04h	補助出力	56
21h	05h	プリンタ出力	56
21h	06h	直接コンソール入出力　【注】	57
21h	07h	フィルタされない文字入力	57
21h	08h	文字入力	57
21h	09h	文字列の出力　【注】	58
21h	0Ah	バッファ付き入力	58
21h	0Bh	入力ステータス入手	58
21h	0Ch	バッファのフラッシュと入力	59
21h	0Dh	ディスクリセット	59
21h	0Eh	デフォルトディスクの選択	59
21h	19h	デフォルトディスクの入手	60

【注】FM TOWNSではコンソール出力はされないことに注意

割込み番号	ﾌｧﾝｸｼｮﾝ番号	ファンクション概要	ﾍﾟｰｼﾞ
21h	1Ah	ディスク転送アドレスの設定	60
21h	1Bh	デフォルトディスクサイズ情報の入手	60
21h	1Ch	ディスクサイズ情報の入手	61
21h	2Ah	システム日付の入手	62
21h	2Bh	システム日付の設定	62
21h	2Ch	システム時間の入手	62
21h	2Dh	システム時間の設定	63
21h	2Eh	ベリファイフラグの設定	63
21h	2Fh	ディスク転送アドレスの入手	63
21h	30h	MS-DOSとDOS-Extenderのバージョンの入手	64
21h	31h	メモリ常駐（ＴＳＲ）プログラム指定	64
21h	32h	MS-DOSディスクブロックの入手	65
21h	33h	CRTL/Cチェックフラグの入手と設定	66
21h	35h	割込みベクタの入手	66
21h	36h	ディスク使用可能スペースの入手	66
21h	38h	国情報の入手または設定	67
21h	39h	サブディレクトリの作成	67
21h	3Ah	サブディレクトリの削除	68

割込み番号	ﾌｧﾝｸｼｮﾝ番号	ファンクション概要	ﾍﾟｰｼﾞ
21h	3Bh	カレントディレクトリの設定	68
21h	3Ch	ファイルの作成	69
21h	3Dh	ファイルのオープン	70
21h	3Eh	ファイルのクローズ	70
21h	3Fh	ファイルまたはデバイスの読出し	71
21h	40h	ファイルまたはデバイスへの書込み	71
21h	41h	ファイルの削除	72
21h	42h	ファイルのシーク	72
21h	43h	ファイル属性の入手または設定	73
21h	44h	デバイス入出力の制御	73
21h	45h	ハンドルの二重化	75
21h	46h	ハンドルの強制的な二重化	76
21h	47h	カレントディレクトリの入手	76
21h	48h	セグメントの割当て	77
21h	49h	セグメントの解放	77
21h	4Ah	セグメントのメモリ割当ての変更	78
21h	4Bh	プログラムの実行	79
21h	4Ch	プログラムの終了	80

割込み番号	ﾌｧﾝｸｼｮﾝ番号	ファンクション概要	ﾍﾟｰｼﾞ
21h	4Dh	子プログラムのリターンコードの入手	80
21h	4Eh	最初に一致するファイルの探索	80
21h	4Fh	次に一致するファイルの探索	81
21h	54h	ベリファイフラグの入手	82
21h	56h	ファイルのリネーム	82
21h	57h	ファイルの日付と時間の入手または設定	82
21h	58h	LOW メモリの割当て方式の入手または設定	83
21h	59h	拡張エラー情報の入手	84
21h	5Ah	テンポラリファイルの作成	84
21h	5Bh	ファイルの作成	85
21h	5Ch	ファイルのロック	85
21h	62h	プログラムセグメントプレフィックス(PSP) の入手	86
22h	なし	終了ルーチンのアドレス	86
23h	なし	CRTL/Cハンドラ	86
24h	なし	クリティカルエラーハンドラ	87
25h	なし	ディスクの絶対モードでの読込み	87
26h	なし	ディスクの絶対モードでの書込み	88
27h	なし	終了後のメモリ常駐プログラム	88

割込み番号	ﾌｧﾝｸｼｮﾝ番号	ファンクション概要	ﾍﾟｰｼﾞ
2Fh	なし	プリントスプーラ	88
2Fh	00h	インストール情報の入手	89
2Fh	01h	ファイルのプリント	89
2Fh	02h	待ち行列からのファイルの削除	90
2Fh	03h	全ファイルの取消し	90
2Fh	04h	ステータス読込みの保留	90
2Fh	05h	保留状態の終了	91

INT 20h　　　プログラム終了

入力　　　　なし

出力　　　　なし

このプログラム終了コールは、DOS-ExtenderによってINT21hファンクション4Ch 終了コールに変換される。この時、FFh のリターンコードがレジスタALにロードされる。

INT 21h　　　MS-DOSシステムサービス

INT 21h のコールは、MS-DOSシステムサービスを要求する場合に使用される。希望のシステムファンクションの番号が AH レジスタによって渡される。正しく動作するほとんどのDOS プログラムは、このMS-DOSインターフェイスを使用する。

INT 21h
AH=00h　　　プログラム終了

入力　　　　なし

出力　　　　なし

このプログラム終了コールは、DOS-Extenderによって、INT 21h ファンクション4C終了コールに変換される。リターンコードFFh がALレジスタにロードされる。

INT 21h
AH=01h　　　文字入力とエコー

入力　　　　なし

出力　　　　AL=8ビット文字

標準入力装置（通常、キーボード）から文字を読み込み、それを標準出力装置（通常、ディスプレイ）に表示する。入力バッファに文字のない場合、この関数はすぐに戻らずに、文字が入力されるまで待つ。

INT 21h

AH=02h　　　文字出力

入力　　　DL=8ビット文字

出力　　　なし

指定された文字を標準出力装置（通常、ディスプレイ）に出力する。

INT 21h

AH=03h　　　補助入力

入力　　　なし

出力　　　AL=8ビット文字

標準補助装置（通常、シリアル通信ポート１）から文字を読み込む。

INT 21h

AH=04h　　　補助出力

入力　　　DL=8ビット文字

出力　　　なし

指定された文字を標準補助装置（通常、シリアル通信ポート１）に出力する。

INT 21h

AH=05h　　　プリンタ出力

入力　　　DL=8ビット文字

出力　　　なし

標準リスト装置（通常、パラレルポート１）に文字を出力する。

INT 21h
AH=06h　　　直接コンソール入出力

入力　　　DL=FFh（文字を読み込む場合）
　=00h-FEh（指定された文字コードを出力する場合）

出力　　　入力要求で文字が入力バッファにある時：
　Zero Flag=クリア
　AL=8ビット文字
入力要求で文字が入力バッファにない時：
　Zero Flag=セット
文字出力の時：
　なし

標準入力装置（通常キーボード）から待たずに文字を読み込むか、標準出力装置（通常ディスプレイ）に、FFh 以外の任意の文字を出力する。

INT 21h
AH=07h　　　フィルタされない文字入力

入力　　　なし

出力　　　AL=8ビット文字

標準入力装置（通常キーボード）から入力した文字のエコーをせずに文字を読み込む。入力バッファに文字のない場合入力されるまで待つ。CTRL-CまたはCTRL-BREAK文字の特別な処理は行わない。

INT 21h
AH=08h　　　文字入力

入力　　　なし

出力　　　AL=8ビット文字

標準入力装置（通常キーボード）からエコーせずに文字を読み込む。入力バッファに文字がない場合入力されるまで待つ。

INT 21h

AH=09h　　文字列の出力

入力	DS:EDX＝出力する文字列のポインタ
出力	なし

指定された文字列（末尾の文字はASCII の‘$’でなければならない）を標準出力装置（通常画面）に出力する。

INT 21h

AH=0Ah　　バッファ付き入力

入力	DS:EDX＝データを読み込むバッファのポインタ
出力	なし

1行分の入力を標準入力装置（通常キーボード）から読み込みそれをバッファに入れる。このファンクションを呼出すプログラムはバッファの先頭1バイトにバッファの最大数を設定する必要がある。2バイト目は実際に入力されたデータのバイト数（復帰文字は除く）が設定され、実際のデータは3バイト目移行に設定される。

INT 21h

AH=0Bh　　入力ステータス入手

入力	なし
出力	AL=00h（文字がない） =FFh（文字がある）

標準入力装置（通常キーボード）のステータスを返す。

INT 21h

AH=0Ch　　バッファのフラッシュと入力

入力	AL＝呼出す入力ファンクションの番号 DS：EDX ＝バッファのポインタ（ファンクション0Ah の場合）
出力	AL＝読み込まれた文字（ファンクション0Ah 以外）

入力バッファをフラッシュし、次に指定されたMS-DOSのファンクション（01h,06h,07h,08h または0Ah)を実行する。

INT 21h

AH=0Dh　　ディスクリセット

入力	なし
出力	なし

ディスクのすべてのファイルバッファをフラッシュする。

INT 21h

AH=0Eh　　デフォルトディスクの選択

入力	DL＝ドライブ番号（0=A:,1=B: など）
出力	AL＝システムの論理ドライブの数

ディスク入出力に用いられるデフォルトのディスクを選択し、システムの論理ドライブの総数を返す。

INT 21h

AH=0Fh～17h　FCB を使用したファイル入出力

DOS-Extenderは FCB（ファイル制御ブロック）を使用するMS-DOSファイル入出力ファンクションをサポートしない。代わりにハンドルベースのファイル入出力機能を使用しなければならない。

INT 21h

AH=19h　　　デフォルトディスクの入手

入力	なし
出力	AL＝ドライブ番号（0=A:, 1=B: など）

ディスク入出力用のカレントのデフォルトディスクを返す。

INT 21h

AH=1Ah　　　ディスク転送アドレスの設定

入力	DS:EDX= データバッファのポインタ
出力	なし

ディスク転送アドレス（DTA)を指定されたバッファに設定する。このバッファは、最低限、使用する機能にとって十分な大きさがなくてはならない。このバッファを使用するファンクションは、4Eh(最初に一致するファイルの探索）と4Fh(次に一致するファイルの探索）だけである。これらの関数は、データバッファ内の43バイトを使用する。

呼出し元プログラムの DTAバッファがLOW のメモリ内にない場合があるので、DOS-Extenderは別の 128バイト DTAバッファを持っている。 DTAバッファを使用するシステムコールが行われると、DOS-Extenderは、まず DOSを呼出し DTAバッファをその内部バッファに設定する。ファンクションがデータをバッファで DOSに渡す場合には、呼出し元プログラムの DTAバッファから内部データバッファにデータをコピーする。その後、指定されたMS-DOSシステムコールを行う。最後に、ファンクションがデータを DTAバッファに返す場合そのデータは呼出し元プログラムの DTAバッファにコピーされる。

プログラムが最初にロードされる時、デフォルトとしてプログラムの PSP内のオフセット80h から始まる 128バイトの領域が DTAバッファに指定される。

INT 21h
AH=1Bh　　　デフォルトのディスクサイズ情報の入手

入力　　　なし

出力　　　AL= セクタ数／クラスタ
DS：EBX= FAT識別バイトのポインタ
CX= 物理セクタのサイズ（単位バイト）
DX= クラスタ数／ディスク

現在のデフォルトディスクドライブのサイズ情報を返す。

INT 21h
AH=1Ch　　　ディスクサイズ情報の入手

入力　　　DL= ドライブ番号（0=現在の省略値、1=A:, 2=B:など）

出力　　　AL= セクタ数/クラスタ
DS：EBX= FAT識別バイトのポインタ
CX= 物理セクタのサイズ（単位バイト）
DX= クラスタ数/ディスク

指定されたディスクドライブのサイズ情報を返す。

INT 21h
AH=21h～24h　FCB を用いたファイル入出力

DOS-Extenderは FCB（ファイル制御ブロック）を使用するMS-DOSファイル入出力機能をサポートしないので、代わりに、ハンドルベースファイル入出力関数を使用しなければならない。

INT 21h
AH=25h　　　割込みベクタの設定

MS-DOSの割込みベクタ入手と設定のファンクションに代わって、DOS-Extenderシステムコールが使用される。これらのシステムコールについては、付録Eを参照。

INT 21h

AH=26h　PSP の作成

DOS-Extenderはこの旧式のファンクションをサポートしない。

INT 21h

AH=27h～29h　FCB を用いたファイル入出力

DOS-Extenderは FCB（ファイル制御）を使用するMS-DOSファイル入出力ファンクションをサポートしないので、代わりに、ハンドルベースファイル入出力ファンクションを使用する必要がある。

INT 21h

AH=2Ah　システム日付の入手

入力　なし

出力　AL= 曜日（0=日曜、1=月曜など）
CX= 年（1980-2099）
DH= 月（1-12）
DL= 日（1-31）

現在のシステム日付を返す。

INT 21h

AH=2Bh　システム日付の設定

入力　CX= 年（1980-2099）
DH= 月（1-12）
DL= 日（1-31）

出力　AL=00h（日付が設定された）
=FFh（日付が無効である）

システム日付を指定された年月日に設定する。

INT 21h
AH=2Ch　　システム時間の入手

入力	なし
出力	CH= 時 (0-23) CL= 分 (0-59) DH= 秒 (0-59) DL= 1/100 秒 (0-99)

現在のシステム時間を返す。

INT 21h
AH=2Dh　　システム時間の設定

入力	CH= 時 (0-23) CL= 分 (0-59) DH= 秒 (0-59) DL= 1/100 秒 (0-99)
出力	AL=00h (時間が設定された) =FFh (時間が無効である)

システム時間を指定された時間に設定する。

INT 21h
AH=2Eh　　ベリファイフラグの設定

入力	AL=00h (ベリファイフラグをオフにする場合) =01h (ベリファイフラグをオンにする場合)
出力	なし

システムベリファイフラグ（ディスクに書き込まれたデータを自動的に検証するためのフラグ）を、指定されたとおりに設定する（この動作で、性能が低下する）。

INT 21h
AH=2Fh　　　ディスク転送アドレスの入手

入力　　　なし

出力　　　ES:EBX=DTAバッファのポインタ

プログラム用の現在のディスク転送アドレス（DTA)バッファのポインタを返す。

INT 21h
AH=30h　　　MS-DOSとDOS-Extenderのバージョン番号の入手

入力　　　EBX="PHAR"（MS-DOSバージョンに加えて、DOS-Extenderバージョンを入手する場合）
=他の任意の値（MS-DOSバージョンだけ入手する場合）

出力　　　AL = MS-DOS メジャーバージョン番号（２進）
AH = MS-DOS マイナーバージョン番号（２進）
EBX<16-31 ビット> = "DX"(DOS-Extender用)
DOS-Extenderのバージョンが要求された時は、以下の値が返る。
BL = DOS-Extender メジャーバージョン番号（ASCII)
BH = DOS-Extender マイナーバージョン番号（ASCII)
EAX<16-23 ビット>= DOS-Extender バージョンレターサブコード(ASCII)
またはASCII 空白文字
EBX<24-31 ビット>= ASCII空白文字
ECX = 環境
= "DOS" ならMS-DOSのみの環境
EDX = 0

このファンクションはMS-DOSバージョン番号とDOS-Extenderの存在を示すサインとをEAX レジスタの上位ワードに返す。加えてDOS-Extenderバージョン番号が要求された場合、バージョン番号がEBX に返され、ECX とEDX は将来の拡張に備えて確保される。DOS-Extenderバージョン番号はASCII で返される。その形式は、EBX レジスタの内容をメモリに格納し、それをASCII文字列として操作できる形式である。たとえば、DOS-Extenderのバージョン1.2 は、文字列 "12" をEBX に返す。バージョンレターサブコードは、DOS-Extenderの内部インクリメンタルリリースにだけ使用される。

INT 21h
AH=31h　　メモリ常駐(TSR) プログラム指定

入力　　AL =リターンコード

出力　　なし

この機能では制御はDOS に戻るが，プログラムおよびDOS-Extenderはメモリ中に常駐されたままで、オープンされたファイルもそのままオープンされたままの状態になる。TSR プログラムはLOW/HIGHメモリとも別のプログラムで利用できるようにできるだけ解放する必要がある。TSR プログラムの詳細は8.8 節を参照のこと。

INT 21h
AH=32h　　MS-DOSディスクブロックの入手

入力　　DL= ドライブ番号（0=省略値、1=A:など）

出力　　ドライブが存在する場合：
AL=00h
DS:EBX= MS-DOSディスクブロックのポインタ
ドライブ番号が無効の場合：
AL=FFh

指定されたドライブのMS-DOSディスクブロックのポインタを返す。ディスクブロックは以下の情報がある。

ｵﾌｾｯﾄ	ｻｲｽﾞ	内　　容
00h	BYTE	ドライブ番号（0=A:, 1=B:など）
01h	BYTE	ドライバ内のユニット
02h	WORD	バイト数/ セクタ
04h	BYTE	セクタ数/ クラスタから１を引いた値
05h	BYTE	クラスタからセクタへのシフト
06h	WORD	確保されたセクタの数
08h	BYTE	ファイル割当てテーブルのエントリ数
09h	WORD	ルートﾃﾞｨﾚｸﾄﾘのエントリ数
0Bh	WORD	クラスタ２のセクタ番号
0Dh	WORD	クラスタの数に１を加えた値
0Fh	BYTE	FAT 用のセクタ
10h	WORD	ﾃﾞｨﾚｸﾄﾘのセクタ番号
12h	DWORD	ドライブヘッダのﾘｱﾙﾓｰﾄﾞアドレス
16h	BYTE	媒体ﾃﾞｨｽｸﾘﾌﾟﾀ バイト
17h	BYTE	ディスクがアクセスされた場合ゼロ
18h	DWORD	次のDOS ディスクブロックのﾘｱﾙﾓｰﾄﾞアドレス

INT 21h

AH=33h　　CTRL/Cチェックフラグの入手と設定

入力　　フラグを入手する場合：

AL=00h

フラグを設定する場合：

AL=01h

DL=00h（CTRL/Cチェックをオフにする）

=01h（CTRL/Cチェックをオンにする）

出力　　DL=00h（CTRL/Cチェックがオフの場合）

=01h（CTRL/Cチェックがオンの場合）

MS-DOS CTRL/C チェックフラグのステータスの入手または設定する。

INT 21h

AH=35h　　割込みベクタの入手

MS-DOSの割込みベクタ入手と設定の機能は、386 | DOS-Extenderシステムコールに置き換えられる。これらのシステムコールについては、付録Eを参照のこと。

INT 21h

AH=36h　　ディスク使用可能スペースの入手

入力　　DL＝ドライブ番号（0=現在の省略値、1A=:, 2=B:など）

出力　　ドライブ番号が無効な場合：

AX＝FFFFh

ドライブ番号が有効な場合：

AX＝セクタ数/ クラスタ

BX＝使用可能なディスククラスタの数

CX＝バイト数/ セクタ

DX＝クラスタ数/ ディスク

ディスクサイズ情報と、ディスクの使用可能量を返す。

INT 21h

AH=38h　　国情報の入手または設定

入力　　AL＝国の選択

＝00h （現在の国の場合）

＝01h-FEh(255 未満の国コードの場合)

＝FFh (BX 内の国コードの場合)

BX＝国コード(255以上の国コードの場合)

現在の国情報を入手する場合：

DS:EDX＝34バイトバッファのポインタ

現在の国を設定する場合：

EDX＝FFFFFFFFh

出力　　正常終了：

キャリーフラグ＝クリア

現在の国情報を入手する場合：

DS:EDXのバッファに入れられる

エラー終了：

キャリーフラグ＝セット

AX＝エラーコード

＝２の時、国コードが無効である。

MS-DOSによって使用されている現在の国を入手または設定する。バッファに格納されるデータの形式は、参考文献－３を参照のこと。

INT 21h

AH=39h　　サブディレクトリの作成

入力　　DS:EDX＝末尾がゼロのパス名のポインタ

出力　　正常終了：

キャリーフラグ＝クリア

エラー終了：

キャリーフラグ＝セット

AX＝エラーコード

＝３：パスがなかった

＝５：アクセスが拒否された

指定された位置にサブディレクトリを作成する。

INT 21h

AH=3Ah　　サブディレクトリの削除

入力	DS:EDX＝末尾がゼロのパス名のポインタ
出力	正常終了： キャリーフラグ＝クリア エラー終了： キャリーフラグ＝セット AX＝エラーコード ＝3：パスがなかった ＝5：アクセスが拒否されたかディレクトリが空である ＝16：カレントディレクトリである

指定されたサブディレクトリを削除する。このディレクトリは空でなくてはならない。また、カレントディレクトリであってはならない。

INT 21h

AH=3Bh　　カレントディレクトリの設定

入力	DS:EDX＝末尾がゼロのパス名のポインタ
出力	正常終了： キャリーフラグ＝クリア エラー終了： キャリーフラグ＝セット AX＝エラーコード ＝3：パスがなかった

現在のカレントディレクトリを指定されたディスクドライブとパスに設定する。

INT 21h
AH=3Ch　　　ファイルの作成

入力　　　CX＝ファイル属性
　＝０：通常ファイル
　＝１：読出し専用ファイル
　＝２：隠しファイル
　＝４：システムファイル
DS:EDX＝末尾がゼロのファイル名のポインタ

出力　　　正常終了：
　キャリーフラグ＝クリア
　AX＝ファイルハンドル
エラー終了：
　キャリーフラグ＝セット
　AX＝エラーコード
　　＝３：パスがなかった
　　＝４：オープンファイルが多過ぎる
　　＝５：アクセスが拒否された

現在のカレントディレクトリまたは指定されたディレクトリにファイルを作成し、以降のファイル入出力コールで使用されるファイルハンドルを返す。ファイルが既に存在している場合には、そのファイルがオープンされファイルの大きさを０にする。

INT 21h

AH=3Dh　　　ファイルのオープン

入力　　　AL＝アクセスとファイルシェアリング（参考文献－３を参照）
　　　　　DS:EDX＝末尾がゼロのファイル名のポインタ

出力　　　正常終了：
　　　　　　キャリーフラグ＝クリア
　　　　　　AX＝ファイルハンドル
　　　　　エラー終了：
　　　　　　キャリーフラグ＝セット
　　　　　　AX＝エラーコード
　　　　　　　＝１：ファイルシェアリングコードが無効
　　　　　　　＝２：ファイルがない
　　　　　　　＝３：パスがない
　　　　　　　＝４：オープンファイルが多過ぎる
　　　　　　　＝５：アクセスが拒否された
　　　　　　　＝12：ファイルアクセスコードが無効

現在のカレントディレクトリまたは指定されたディレクトリ中の指定されたファイルをオープンし、以降のファイル入出力コールで使用されるファイルハンドルを返す。

INT 21h

AH=3Eh　　　ファイルのクローズ

入力　　　BX＝ファイルハンドル

出力　　　正常終了：
　　　　　　キャリーフラグ＝クリア
　　　　　エラー終了：
　　　　　　キャリーフラグ＝セット
　　　　　　AX＝エラーコード
　　　　　　　＝６：ハンドルが無効

指定されたファイルハンドルで以前オープンされたファイルをクローズする。

INT 21h

AH=3Fh　　ファイルまたはデバイスの読出し

入力	BX＝ファイルハンドル ECX＝読み出すバイト数 DS:EDX＝データを読み込むバッファのポインタ
出力	正常終了： 　キャリーフラグ＝クリア 　EAX＝実際に読み込まれたバイト数 エラー終了： 　キャリーフラグ＝セット 　AX＝エラーコード 　　＝５：アクセスが拒否された 　　＝６：ハンドルが無効

指定されたファイルハンドルで、既にオープンされたファイルまたはデバイスからデータを読み出す。読み出し中にファイルの終りに達した場合、要求以下のデータ（バイト数）が返されることがある。

INT 21h

AH=40h　　ファイルまたはデバイスの書込み

入力	BX＝ファイルハンドル ECX＝書き込むバイト数 DS:EDX＝書き込みデータを格納するバッファのポインタ
出力	正常終了： 　キャリーフラグ＝クリア 　EAX＝実際に書き込まれたバイト数 エラー終了： 　キャリーフラグ＝セット 　AX＝エラーコード 　　＝５：アクセスが拒否された 　　＝６：ハンドルが無効

指定されたファイルハンドルで、以前にオープンされたファイルまたはデバイスにデータを書き込む。ファイルのあるデバイスがオーバフローした場合、実際に書き込まれたデータが要求に満たないことがある。

INT 21h

AH=41h　　　ファイルの削除

入力	DS:EDX＝末尾がゼロのファイル名のポインタ
出力	成功した場合： 　キャリーフラグ＝クリア 失敗した場合： 　キャリーフラグ＝セット 　AX＝エラーコード 　　＝２：ファイルがない 　　＝５：アクセスが拒否された

指定されたファイルが削除される。

INT 21h

AH=42h　　　ファイルのシーク

入力	AL＝オフセットのオリジン 　＝00h：オフセットがファイルの始めからの場合 　＝01h：オフセットが現在のファイルポインタからの場合 　＝02h：オフセットがファイルの終りからの場合 BX＝ファイルハンドル CX＝オフセット値の上位ワード DX＝オフセット値の下位ワード
出力	正常終了： 　キャリーフラグ＝クリア 　DX＝新しいファイルポインタの上位ワード 　AX＝新しいファイルポインタの下位ワード エラー終了： 　キャリーフラグ＝セット 　AX＝エラーコード 　　＝１：AL内の原点コードが無効 　　＝６：ファイルハンドルが無効

ファイル読出し／書込みポインタを位置づける。ダブルワードのオフセットが設定される。

INT 21h

AH=43h　　ファイル属性の入手または設定

入力　　AL＝00h ：ファイル属性を入手する場合
　＝01h ：ファイル属性を設定する場合
DS:EDX＝末尾がゼロのファイル名のポインタ
ファイル属性を設定する場合：
CX＝新しい属性
ビット0 ＝読出し専用
ビット1 ＝隠し（hidden）
ビット2 ＝システム
ビット5 ＝アーカイブ

出力　　正常終了：
キャリーフラグ＝クリア
ファイル属性を入手する場合：
CX＝属性
エラー終了：
キャリーフラグ＝セット
AX＝エラーコード
＝1 ：AL内のファンクションコードが無効
＝2 ：ファイルがない
＝3 ：パスがない
＝5 ：属性を変更できない

指定されたファイルの属性の読出しまたは変更を行う。

INT 21h

AH=44h　　デバイス入出力の制御

入力　　デバイス情報を入手する場合：
AL＝00h
BX＝ファイルまたはデバイスハンドル
デバイス情報を設定する場合：
AL＝01h
BX＝ファイルまたはデバイスハンドル
DX＝デバイス情報

キャラクタデバイス制御チャネルから読み出す場合：

AL＝02h

BX＝デバイスハンドル

CX＝読出しバイト数

DS:EDX＝データバッファのポインタ

キャラクタデバイス制御チャネルに書き込む場合：

AL＝03h

BX＝デバイスハンドル

CX＝書込みバイト数

DS:EDX＝データバッファのポインタ

ブロックデバイス制御チャネルから読み出す場合：

AL＝04h

BL＝デバイス番号（0=デフォルト、1=A:など）

CX＝読出しバイト数

DS:EDX＝データバッファのポインタ

ブロックデバイス制御チャネルに書き込む場合：

AL＝05h

BL＝ドライブ番号（0=デフォルト、1=A:など）

CX＝書込みバイト数

DS:EDX＝データバッファのポインタ

入力ステータスを入手する場合：

AL＝06h

BX＝ファイルまたはデバイスハンドル

出力ステータスを入手する場合：

AL＝07h

BX＝ファイルまたはデバイスハンドル

ブロックデバイスが変更可能かどうか調べる場合：

AL＝08h

BL＝ドライブ番号（0=デフォルト、1=A:など）

ドライブがローカルかリモートかを調べる場合：

AL＝09h

BL＝ドライブ番号（0=デフォルト、1=A:など）

ハンドルがローカルかリモートかを調べる場合：

AL= 0Ah

BX＝デバイスハンドル

共用リトライカウトを変更する場合：

AL＝0Bh

BX＝デバイスハンドル

CX＝遅延ループの数

DX＝リトライの数

出力　　　　　正常終了：
キャリーフラグ＝クリア
ファンクション00h:
DX＝デバイス情報
ファンクション02h-05h:
AX＝転送されたバイト数
ファンクション06h-07h:
AL＝00h（デバイスがレディでない）
＝FFh（デバイスがレディである）
ファンクション08h:
AX＝０（取外し可能媒体）
＝１（固定媒体）
エラー終了：
キャリーフラグ＝セット
AX＝エラーコード
＝１：AL内のファンクションが無効な
＝５：アクセスが拒否された
＝６：ハンドルが無効
＝13：データが無効
＝15：ドライブ番号が無効

このファンクションは、デバイスドライバパラメータを入手または変更する場合に使用される。ファンクション00h と01h が使用するデバイス情報ワードの形式は、キャラクタデバイスとブロックデバイスとでは異なる。参考文献－３を参照のこと。

INT 21h
AH=45h　　　　ハンドルの二重化

入力　　　　　BX＝ファイルまたはデバイスハンドル

出力　　　　　正常終了：
キャリーフラグ＝クリア
AX＝新しいハンドル
エラー終了：
キャリーフラグ＝セット
AX＝エラーコード
＝４：オープンされているファイルが多過ぎる
＝６：ハンドルが無効

このファンクションは、指定されたハンドルと同じファイルまたはデバイスに関連する新しいハンドルを作成する。

INT 21h
AH=46h　　　　ハンドルの強制的な二重化

入力　　　　BX＝コピーするハンドル
　　　　　　CX＝新しいハンドル

出力　　　　正常終了：
　　　　　　　キャリーフラグ＝クリア
　　　　　　エラー終了：
　　　　　　　キャリーフラグ＝セット
　　　　　　　AX＝エラーコード
　　　　　　　　＝4 ：オープンされているファイルが多過ぎる
　　　　　　　　＝6 ：ハンドルが無効

新しいハンドルは元のハンドルと同じファイルまたはデバイスに関連すると強制的にみなされる。新しいハンドルがオープンされているファイルに関連する場合、ハンドルをコピー前にファイルがクローズされる。

INT 21h
AH=47h　　　　カレントディレクトリの入手

入力　　　　DL＝ドライブ番号（0=省略時、1=A:など）
　　　　　　DS:ESI＝ディレクトリ名を格納する64バイトバッファのポインタ

出力　　　　正常終了：
　　　　　　　キャリーフラグ＝クリア
　　　　　　エラー終了：
　　　　　　　キャリーフラグ＝セット
　　　　　　　AX＝エラーコード
　　　　　　　　＝15：ドライブ番号が無効

指定されたディスクドライブの現在のカレントディレクトリ（末尾がゼロの）の名前を返す。

INT 21h

AH=48h　　セグメントの割当て

入力	EBX=割り当てる4KBメモリページの数
出力	正常終了： キャリーフラグ＝クリア AX＝割り当てられたセグメントのセグメントセレクタ エラー終了： キャリーフラグ＝セット AX＝エラーコード ＝8：メモリが不足しているか、LDTが最大サイズ EBX=メモリの使用可能なページの数

ローカルディスクリプタテーブル(LDT)のセグメントディスクリプタを割り当て、セグメント参照用のセグメントセレクタを返す。セグメントは常に、書き込み可能なデータセグメントに作成される。ゼロページのメモリ割り当ては可能である。これはプログラム自身の要求に合うように変更可能なLDTディスクリプタを割り当てるためである。

INT 21h

AH=49h　　セグメントの解放

入力	ES＝解放するセグメントのセグメントセレクタ
出力	正常終了： キャリーフラグ＝クリア ES= 0000h(ヌルセグメントセレクタ) エラー終了： キャリーフラグ＝セット EAX=エラーコード =9：セグメントセレクタが無効

指定されたセグメントを解放し、そのセグメントに割り当てられたメモリページのすべてを解放する。セグメントはローカルディスクリプタテーブル(LDT)の中になくてはならずDOS-Extenderによって設定されたハードワイヤドなセグメントであってはなない。（ただし、元のコードとデータセグメントを除く）。コードまたはデータセグメントが解放される場合、プログラムはまずそのスタックを別のセグメントに移動してから、別のセグメントから実行していなければならない。そうしないとシステムは暴走する。

コードセグメントとデータセグメントは別名であり、すべて同じプログラムセグメントにあるとして処理され、一方が解放されれば、両方のLDT ディスクリプタが解放される。

INT 21h

AH=4Ah　　　　セグメントのメモリ割当ての変更

入力　　　ES＝変更するセグメントのセグメントセレクタ
　　　　　EBX=新しいセグメントサイズ（4キロバイトページ単位）

出力　　　正常終了：
　　　　　　キャリーフラグ＝クリア
　　　　　エラー終了：
　　　　　　キャリーフラグ＝セット
　　　　　　EAX=エラーコード
　　　　　　　＝8：メモリが不足している
　　　　　　　＝9：セグメントセレクタが無効
　　　　　　EBX=使用可能なメモリのページ数

セグメントに割り当てられたメモリサイズを変更し、LDT 内のセグメントリミットを変更する。セグメントはローカルディスクリプタテーブル(LDT) 内になければならず、またDOS-Extenderが設定するハードワイヤドなセグメントであってはならない（ただし、元のコード、データおよびスタックセグメントは除く）。
DOS-Extenderは新しいセグメントリミットが正しいことを確認するためにセグメントレジスタを再ロードする。
コードセグメントとデータセグメントは別名であり、両方とも同一プログラムセグメントに属するとみなされ、一方が変更されれば両方のセグメントリミットがLDT 内で変更される。
このファンクションはプログラムに不要なメモリを解放して他のプログラムが使用できるようにする場合に役立つ。デフォルトでは、マシン内のすべてのメモリがプログラムのロード時に割り当てられる。したがって、正しく動作するプログラムは不要なメモリを初期設定時に解放する。最初のプログラムサイズに関する情報は、PSP にある。本文第3.3節を参照のこと。
このファンクションによってセグメントが拡大した場合、LDT セグメントディスクリプタ中のセグメントのリニアベースアドレスが変わることがある。この副作用はアプリケーションにはわからない。

INT 21h

AH=4Bh　　　プログラムの実行

入力	AL=00h（プログラムをロードし実行する場合） 　＝他の値はすべて無効 ES:EBX＝パラメタブロックのポインタ DS:EDX＝末尾がゼロのプログラムファイル名のポインタ	
出力	正常終了： 　キャリーフラグ＝クリア 　レジスタはすべて変更されない。 エラー終了： 　キャリーフラグ＝セット 　EAX ＝エラーコード 　　＝1 ：AL内の関数コードが無効 　　＝2 ：ファイルがないか、パスが無効 　　＝5 ：アクセスが拒否された 　　＝8 ：プログラムロード用のメモリが不足 　　＝10：環境が無効 　　＝11：ファイル形式が無効 他のすべてのレジスタは変更されない。	

このファンクションは他のリアルモードプログラムまたは386 ネイティブモードプログラムをロードして実行する場合に使用される。子プログラムのロードができるように、十分なメモリを確保しなければならない。本文の第 3.8節を参照のこと。

ES=EBXの示すパラメタブロックの形式を以下に示す。

DWORD	環境文字列のオフセット
WORD	環境文字列のセグメントセレクタ
DWORD	コマンドテール文字列のオフセット
WORD	コマンドテール文字列のセグメントセレクタ

ヌル（ゼロ）セグメントレジスタが環境文字列用のセグメントセレクタとして渡された場合、親プログラムの環境ブロックがコピーされて子プログラムに割り当てられる。これ以外では環境文字列は末尾ゼロの文字列であり、ブロック全体はもう一つのゼロバイトで終わる。

コマンドテール文字列は、カウントバイトとそれに続く（キャリッジリターン文字で終る)ASCII文字列から構成される。カウント値には、カウントバイト自体または終了のためのキャリッジリターン文字は含れず、ASCII 文字列中のバイト数を示すだけである。

INT21h
AH=4ch　　　　プログラムの終了

入力	AL＝リターンコード
出力	なし

プログラムを終了し、制御をMS-DOSまたは親プログラムに戻す。プログラムの終了に先立ち、すべてのオープン済のファイルがクローズされファイルバッファの内容がディスクに書き出される。

INT21h
AH=4Dh　　　　子プログラムのリターンコードの入手

入力	AH＝出口コード ＝00h(通常の出口の場合) ＝01h(CTRL/Cを使って終了した場合) ＝02h(重大デバイスエラーの発生した場合) ＝03h(終了後も常駐した場合) AL＝子プログラムによって渡されたリターンコード

EXECシステムコール（４Bh）が正常終了した後、この機能を使用して子プログラムのリターンコードと子プログラムの終了の状態を入手できる。

INT21h
AH=4Eh　　　　最初に一致するファイルの探索

入力	CX＝ファイル属性 DS:EDX＝末尾がゼロのファイルパスのポインタ
出力	正常終了： キャリーフラグ＝クリア エラー終了： キャリーフラグ＝セット AX＝エラーコード ＝２：パスが無効 ＝18：一致するディレクトリエントリがない

ファイル名が一致し、指定されたファイル属性のサブセットを持つ、指定されたディレクトリ内の最初のファイルを返す。ファイルパスの名前部には'＊'と'？'のワイルドカード文字を使用できる。結果はプログラムのDTA バッファに以下の形式で格納される。

ｵﾌｾｯﾄ	サイズ	意　味
00h	21 BYTEs	MS-DOSのリザーブ
15h	BYTE	ファイル属性
16h	WORD	ファイル時間
18h	WORD	ファイル日付
2Ah	DWORD	ファイルサイズ（単位バイト）
2Eh	13 BYTEs	末尾がゼロのファイル名

INT21h
AH=4Fh　　次に一致するファイルの探索

入力　　なし

出力　　正常終了：
　キャリーフラグ＝クリア
エラー終了：
　キャリーフラグ＝セット
AX＝エラーコード
　＝18：一致するディレクトリエントリがない

このシステムコールが実行される場合、直前のシステムコール：4Ehまたは4Fhの結果がDTA バッファになければならない。DTA バッファ内の結果の格納形式は、システムコール4Ehで述べた形式と同じである。

INT21h
AH=54h　　ベリファイフラグの入手

入力　　なし

出力　　AL＝00h(ベリファイフラグがオフの場合)
　＝01h(ベリファイフラグがオンの場合)

ディスクに書き込まれたデータを自動的にベリファイできるようにするのに使用されるシステムフラグの現在の状態を返す。

INT21h

AH=56h　　　ファイルのリネーム

入力　　　DS:EDX＝末尾がゼロのカレントファイル名のポインタ
ES:EDI＝末尾がゼロの新しいファイル名のポインタ

出力　　　正常終了：
　キャリーフラグ＝クリア
エラー終了：
　キャリーフラグ＝セット
AX＝エラーコード
　＝2：ファイルがない
　＝3：パス名がない
　＝5：アクセスが拒否された
　＝17：デバイスが異なる

ファイル名をつけ直して、同一ディスクの別ディレクトリにそのファイル名を移す（オプション）

INT21h

AH=57h　　　ファイルの日付と時間の入手または設定

入力　　　BX＝ファイルハンドル
日付と時間を入手する場合：
　AL＝00h
日付と時間を設定する場合：
　AL＝01h
　CX＝時間
　DX＝日付

出力　　　正常終了：
　キャリーフラグ＝クリア
日付と時間を入手する場合：
　CX＝時間
　DX＝日付
エラー終了：
　AX＝エラーコード
　　＝1：AL内の関数コードが無効
　　＝6：ハンドルが無効

指定されたファイルの日付と時間を返すかまたは、変更する。返されるデータの形式を以下に示す。

時間のデータの形式

ビット０－４＝秒（２秒に１づつ増加）

ビット５－10＝分

ビット11－15＝時間

日付のデータの形式

ビット０－４＝日

ビット５－８＝月

ビット９－15＝年（1980からの年）

INT21h

AH=58h　　LOW メモリの割当て方式の入手または設定

入力　　割り当て方式を入手する場合：

AL＝00h

割り当て方式を設定する場合：

AL＝01h

BX＝方式コード

＝０：下位のブロックから検索し、最初に見つかったブロックを割り当てる。

＝１：利用可能なブロックを検索し、必要最小の利用可能ブロックを割り当てる。

＝２：上位のブロックから検索し、最初に見つかったブロックを割り当てる。

出力　　正常終了：

キャリーフラグ＝クリア

方式を入手する場合：

AX＝方式コード

エラー終了：

キャリーフラグ＝セット

AX＝エラーコード

＝１：AL内のファンクションコードが無効

このファンクションは、MS-DOSのLOW メモリの割り当て方法を入手または設定する。DOS-Extenderシステムコール25C0h - 25C2h によって実行されるメモリ割当ては、この影響を受ける。この割当て方式は、INT21hファンクション 48h－4Ah で実行されるメモリ割当てとは関係ない。

INT21h

AH=59h　拡張エラー情報の入手

入力　BX＝０

出力　AX＝拡張エラーコード
BH＝エラークラス
BL＝対処方法
CH＝エラー発生場所
CL, DX, SI, DI, BP, DS, ES＝破壊される

前のMS-DOS V2.0 に対して拡張さたエラー情報を返す。エラーコードについては、参考文献－３を参照のこと。

INT21h

AH=5Ah　テンポラリファイルの作成

入力　CX＝ファイル属性
DS:EDX＝末尾がゼロのパス名のポインタ

出力　正常終了：
キャリーフラグ＝クリア
AX＝ファイルハンドル
DS:EDX＝パス名の後に追加されたファイル名
エラー終了：
キャリーフラグ＝セット
AX＝エラーコード
＝３：パスがない場合
＝５：アクセスが拒否された場合

指定されたディレクトリ中にテンポラリファイルを作成し、そのファイル名を返す。DS:EDXの示すバッファはパス名の後に13バイトの空白がなければならない。

INT21h

AH=5Bh　　ファイルの作成

入力　　CX＝ファイル属性
DS:EDX＝末尾がゼロのファイル名のポインタ

出力　　正常終了：
　キャリーフラグ＝クリア
　AX＝ファイルハンドル
エラー終了：
　キャリーフラグ＝セット
　AX＝エラーコード
　　＝3：パスがない
　　＝4：オープンファイルが多過ぎる
　　＝5：アクセスが拒否された場合
　　＝80：ファイルが既に存在する

指定されたディレクトリまたはカレントディレクトリ中に指定されたファイルを作成する。

INT21h

AH=5Ch　　ファイルのロック

入力　　AL＝00h（ファイル領域をロックする場合）
　＝01（ファイル領域をアンロックする場合）
BX＝ファイルハンドル
CX＝領域オフセットの上位ワード
DX＝領域オフセットの下位ワード
SI＝領域サイズの上位ワード
DI＝領域サイズの下位ワード

出力　　正常終了：
　キャリーフラグ＝クリア
エラー終了：
　キャリーフラグ＝セット
　AX＝エラーコード
　　＝1：AL内のファンクションコードが無効

=６：ハンドルが無効
=33：領域の全部または一部が既にロックされている

このファンクションはマルチタスク環境かネットワーク環境で、ファイル領域をロックまたはアンロックするのに使用される。ロックする領域のオフセットと領域サイズは、ともにバイト単位で示される。

INT21h
AH=62h　　プログラムセグメントプレフィクスの入手

入力	なし
出力	BX＝プログラムのセグメントプレフィクスのセグメントセレクタ

プログラムのPSP をマッピングするセグメントのセレクタを返す。

INT　22h　終了ルーチンのアドレス

このファンクションは、リアルモードの割込みベクタで、終了ルーチンのアドレスを得る。一般にMS-DOSのもとで実行中のプログラムを終了させる場合は、INT 21hのファンクション4chをコールする。

INT　23h　CRTL／Cハンドラ

この割込みは、リアルモードでDOS のファンクションをコールする度に、そして、キーボードの入力バッファにCRTL／C キャラクタがあるときにMS-DOSにより起こる。
プログラムで、リアルモードのハンドラ(386|DOS-Extenderシステムのファンクション2505h)による制御か、またはプロテクトモードのハンドラ(386|DOS-Extemderシステム)のファンクション2506h)による制御かのいずれかによりこの割込みが発生する。

INT 24h クリティカルエラーハンドラ

この割込みは、MS-DOSのリアルモードのハンドラ(386 | DOS-Extenderシステムのファンクション2505h)による制御か、またはプロテクトモードのハンドラ(386 | DOS-Extenderシステムのファンクション2506h)による制御かのいずれかにより発生する。

INT 25h ディスクの絶対モードでの読込み

入力　AL= ドライブ番号(0=A:, 1=B:, など)

DOS 3.3 で拡張フォーマットを使用する場合

CX=FFFFHh

DS:FBX=コールパラメータのポインタ

32MB以下のディスクで標準フォーマットを使用する場合

CX=読込むセクタ番号

DX=読込みを開始する論理セクタ番号

DS:EBX=データを読込むバッファのポインタ

出力　正常終了:

キャリーフラグ=クリア

AX, BX, CX, DX, SI, DI, BP= 破壊される

エラー終了:

キャリーフラグ=セット

AX=エラーコード

BX, CX, DX, SI, DI, BP= 破壊される

このファンクションは、論理ディスクセクタを指定してデータを読み込む場合に使用する。DOS3.3のもとで拡張フォーマットを指定すると、そのパラメータはDS:EBXで指定された次の情報を与えます。

DWORD	開始論理セクタ番号
WORD	読込むセクタ番号
DWORD	読込むデータバッファのオフセット
WORD	データバッファのセグメントセレクタ

INT26h　　ディスクの絶対モードでの書込み

入力	AL= ドライブ番号（0＝A：1＝B：など）	
	DOS3.3用の拡張形式を使用する場合:	
		CX=FFFFh
		DS:EBX= コールパラメータが入ったパケットのポインタ
	32MB未満のディスクパーティション用の標準形式を適用する場合:	
		CX= 書き込むセクタ数
		DX= 開始論理セクタ番号
		DS:EBX= 書き込むデータが入っているバッファのポインタ
出力	正常終了:	
		キャリーフラグ= クリア
		AX,BX,CX,DX,SI,DI,BP= 破壊される
	エラー終了:	
		キャリーフラグ= セット
		AX= エラーコード
		BX,CX,DX,SI,DI,BP ＝破壊される

この関数は、データを論理ﾃﾞｨｽｸｾｸﾀ に書き込む。DOS3.3用に定義された拡張形式が使用される場合、DS:EBXの示すﾊﾟﾗﾒｰﾀﾊﾟｹｯﾄ には以下の情報が記憶される。

DWORD	開始論理セクタ番号
WORD	書き込むセクタ数
DWORD	書き込むデータが入ったデータバッファのオフセット
WORD	データバッファのセグメントセレクタ

INT27h　　終了後のメモリ常駐プログラム

386ﾈｲﾃｨﾌﾞﾓｰﾄﾞ の終了後も常駐するﾌﾟﾛｸﾞﾗﾑ はDOS-Extenderによってｻﾎﾟｰﾄされない。この関数コールは、実行されると、DOS-ExtenderによってINT21h関数 4 Chﾀｰﾐﾈｲﾄｺｰﾙ に変換される。このとき、ﾘﾀｰﾝｺｰﾄﾞ FFh がﾚｼﾞｽﾀALにロードされる。

INT2Fh　　プリントスプーラ

INT2Fhコールは、MS-DOSﾌﾟﾘﾝﾄｽﾌﾟｰﾗのｻｰﾋﾞｽを要求する場合に使用される。

INT2Fh
AL=00h　　　インストール情報の入手

入力　　　なし

出力　　　正常終了:
　キャリーフラグ=クリア
　AL=ステータス
　　=00h(スプーラがインストールされていない場合、インストール可)
　　=01h(スプーラがインストールされていない場合、インストール不可)
　　=FFh(スプーラがインストールされている場合)
エラー終了:
　キャリーフラグ=セット
　AX=エラーコード

プリントスプーラのインストール情報を返す。このコールは、他のスプーラコール を行う前に、プリントスプーラの有無を調べるのに使用する必要がある。

INT2Fh
AL=01h　　　ファイルのプリント

入力　　　DS:EDX=7バイトパケットのポインタ

出力　　　正常終了:
　キャリーフラグ=クリア
エラー終了:
　キャリーフラグ=セット
　AX=エラーコード

プリントのための待ち行列に入れるのに、ファイルをスプーラに投入する。パケットはレベルおよびプリントされるファイル名とファイルパスを以下の形式で指定する。

BYTE	レベル
DWORD	末尾がゼロのファイル名文字列のオフセット
WORD	ファイル名文字列のセグメントセレクタ

INT2Fh
AL=02h　　待ち行列からのファイルの削除

入力	DS:EDX＝末尾がゼロのファイル名文字列のポインタ
出力	正常終了: キャリーフラグ＝クリア エラー終了: キャリーフラグ＝セット AX＝エラーコード

指定されたﾌｧｲﾙをﾌﾟﾘﾝﾄ待ち行列から削除する。'＊'と'？'のﾜｲﾙﾄﾞｶｰﾄﾞ文字は、ﾌｧｲﾙ名文字列中に使用することができる。

INT2Fh
AL=03h　　全ファイルの取消し

入力	なし
出力	正常終了: キャリーフラグ＝クリア エラー終了: キャリーフラグ＝セット AX＝エラーコード

現在ﾌﾟﾘﾝﾄ待ち行列にあるすべてのﾌｧｲﾙを取消す。

INT2Fh
AL=04h　　ステータス読み込みの保留

入力	なし
出力	正常終了: キャリーフラグ＝クリア DX＝エラーカウント DS:ESI＝プリント待ち行列のポインタ エラー終了: キャリーフラグ＝セット AX＝エラーコード

ﾌﾟﾘﾝﾄ待ち行列中のジョブを保留して、それらを走査できるようにする。他のｽﾌﾟｰﾗｺｰﾙ を発行すると保留が解除される。ﾌﾟﾘﾝﾄ待ち行列は、64バイトの末尾がゼロのﾌｧｲﾙ名文字列の配列から構成される。最初の欄は現在ﾌﾟﾘﾝﾄ中のﾌｧｲﾙである。待ち行列の終りには、最初のバイトにゼロを持つ欄がある。

INT2Fh

AL=05h　　保留状態の終了

入力　　なし

出力　　正常終了:

　キャリーフラグ＝クリア

エラー終了:

　キャリーフラグ＝セット

　AX＝エラーコード

関数04h によって保留0 されたﾌﾟﾘﾝﾄ待ち行列のジョブを解除する。

付録E

386 | DOS-Extender システムコール

DOS-Extenderシステムコールは、AXレジスタにファンクションコードを入れてソフトウェア割込み21hを使用して行われます。これらのシステムコールは386 ネイティブモードでのみ実行できます。

INT 21h 割込みはMS-DOSシステムコールのためにも使用されます。DOS-ExtenderシステムコールはAHレジスタの25h の値によって識別されます。リアルモードでは、この値は割込みベクトルの設定のMS-DOSシステムコールです。386 ネイティブモードでは割込みベクトルを操作するいくつかのDOS-Extenderシステムコールに置き換えられます。これから説明するDOS-ExtenderのシステムコールのコードはALレジスタで受け渡されます。

すべてのレジスタ（結果を返すのに使用されるレジスタを除く）の値がDOS-Extenderシステムコールの前後にわたって保存されます。DOS-Extenderシステムコールはエラー発生時にはプロセッサキャリーフラグをセットし、正常終了時はキャリーフラグをクリアします。キャリーフラグがセットされて無効な値が返された時は、EAX レジスタに"A5A5A5A5"をセットし他のレジスタは変更されません。

表　DOS-Extenderシステムコール一覧

機能番号	機　能　概　要	ページ
2501h	DOS-Extenderデータ構造体のリセット	94
2502h	386 ネイティブモード割込みベクタの入手	94
2503h	リアルモード割込みベクタの入手	95
2504h	386 ネイティブモード割込みベクタの設定	95
2505h	リアルモード割込みベクタの設定	96
2506h	常に386 ネイティブモードで制御権を獲得するための割込み設定	96
2507h	リアルモードと386 ネイティブモードの割込みベクタの設定	97
2508h	セグメントリニアベースアドレスの入手	97
2509h	リニアアドレスの物理アドレスへの変換	97
250Ah	セグメントの終端での物理メモリのマッピング	98
250Ch	ハードウェア割込みベクタの入手	99
250Dh	リアルモードのリンク情報の入手	99
250Eh	リアルモードプロシジャの呼出し	101
250Fh	386 ネイティブモードアドレスのMS-DOSアドレスへの変換	102
2510h	リアルモードプロシジャの呼出し（レジスタ指定）	102
2511h	リアルモード割込みの発行（レジスタ指定）	103
2512h	デバッグのためのプログラムのロード	104
2513h	セグメントディスクリプタの別名の設定	104

機能番号	機 能 概 要	ページ
２５１４h	セグメント属性の変更	107
２５Ｃ０h	MS-DOSメモリブロックの割当て	102
２５Ｃ１h	MS-DOSメモリブロックの解放	103
２５Ｃ２h	MS-DOSメモリブロック割当ての変更	113

INT21h
AX=2501h　　DOS-Extenderデータ構造体のリセット

入力　　なし

出力　　キャリーフラグ＝クリア

リアルモードと386 ネイティブモード間のモード切換え時に割り当てられたDOS-Extenderデータ構造体が解放される。このシステムコールの目的は１レベル以上モード切換えをした時にプログラムが自身を再スタートできるようにすることである。

DOS-Extenderはモードを切換えるたびにプロセッサレジスタを保存するデータ構造体を割り当てる。加えて、386 ネイティブモードからリアルモードへ切換える時には、スタックメモリバッファがリアルモードのスタック用に割り当てられる。割り当てられたデータ構造体は、モード切換え後に制御権を獲得したコードが終了すると解放され、その結果前のモードに切換えられる。

アプリケーションプログラムが、入れ子になったモード切換えを一つずつ抜けて戻らないで、単に自身の再スタートをするだけであれば、割り当てられたDOS-Extenderデータ構造体は絶対に解放されない。その結果、DOS-Extenderはプログラムの後の方でデータ構造体を使い果たす可能性がある。

典型的な例としては、リアルモードでのMS-DOS重大エラー割込みを引きつぐプログラムがある。このプログラムはリアルモードで制御権を獲得した時、386 ネイティブモードに切換えて再スタートすることにする。再スタート処理の一部として、再スタートルーチンは、データ構造体を解放するためにこのシステムコールを行う必要がある。このシステムコールを行う前に、再スタートルーチンがアプリケーションプログラムが所有するスタックをセグメント内にセットアップすることが重要である。セットアップをしないと、スタックは前のモード切換え時にDOS-Extenderによって割り当てられたスタックメモリバッファ内にあるかもしれない。いったんシステムコールが実行されると、スタックバッファは

再使用が可能になる。これはハードウェア割込みが、同一スタックバッファを再割当てさせられることを意味する。

INT21h
AX=2502h　　386 ネイティブモード割込みベクタの入手

入力　　CL＝入手する割込みの番号

出力　　キャリーフラグ＝クリア
ES:EBX＝386 ネイティブモード割込みハンドラルーチンのアドレス

このシステムコールは386 ネイティブモード割込みベクタを変更する前に、IDT(割込みディスクリプタテーブル）内の386 ネイティブモード割込みベクトルを退避するために使用される。これにより元のベクタは後で復元きる。

INT21h
AX=2503h　　リアルモード割込みベクタの入手

入力　　CL＝入手する割込みの番号

出力　　キャリーフラグ＝クリア
EBX ＝リアルモード割込みハンドラルーチンのアドレス

リアルモードのセグメント：オフセットアドレスは、EBX の上位16ビットにセグメントパラグラフアドレス、下位16ビットにオフセットを入れて返される。このシステムコールはリアルモード割込みベクタのオリジナルの値を、変更する前に退避することに使用される。その結果最初のベクタはプログラムの終了前に復元できる。

INT21h
AX=2504h　　386 ネイティブモード割込みベクタの設定

入力　　CL＝設定する割込みの番号
DS:EDX＝386 ネイティブモード割込みハンドラのアドレス

出力	キャリーフラグ＝クリア

386 ネイティブモードIDT 内の指定された割込み用ベクタは、指定された割込みハンドラルーチンを指すように設定される。IDT ディスクリプタエントリ中のディスクリプタタイプは必ず "386割込みゲート" に設定される。

INT21h
AX=2505h　　リアルモード割込みベクタの設定

入力	CL＝設定する割込みの番号 EBX ＝リアルモード割込みハンドラのアドレス
出力	キャリーフラグ＝クリア

MS-DOSメモリの先頭にあるリアルモード割込みテーブル中の指定されたベクタが、指定された割込みハンドラルーチンを示すように設定される。リアルモードセグメント：オフセットアドレスは、EBX の上位16ビットにセグメントのパラグラフアドレスを、下位16ビットにオフセットを入れて渡される。

INT21h
AX=2506h　　常に386 ネイティブモードで制御権を獲得するための割込み設定

入力	CL＝設定する割込みの番号 DS:EDX＝386 ネイティブモード割込みハンドラのアドレス
出力	キャリーフラグ＝クリア

このファンクションは386 ネイティブモードまたはリアルモードで割込みが発生しても、必ず386 ネイティブモード割込みハンドラが制御権を獲得するように、指定された割込みを引きつぐ。このファンクションは386 ネイティブモード割込みディスクリプタテーブルとリアルモード割込みベクトルテーブルの両方を変更する。したがって、386 ネイティブモードとリアルモード割込みの両方に対する元の割込みベクトルは、このシステムコール実行前に保存し、プログラム終了前に復元しなければならない。

このシステムコールはハードウェア割込みを引きつぐ場合か、リアルモードから386 ネイティブモードへの切換えに使われるソフトウェア割込みを設定する場合に役立つ。

INT21h
AX=2507h　　　リアルモードと386 ネイティブモードの割込みベクタの設定

入力　　　　CL＝設定する割込みの番号
　　　　　　DS:EDX＝386 ネイティブモード割込みハンドラのアドレス
　　　　　　EBX ＝リアルモード割込みハンドラのアドレス

出力　　　　キャリーフラグ＝クリア

このシステムコールはシステムコール：2504h と2505h が行う操作を組み合わせている。これらのシステムコールとの違いは、このファンクションはリアルモードと386 ネイティブモードの割込みベクタの両方を一括して設定することである。つまり、両方のベクタが変更されるまで割込みはできない。これはハードウェア割込みに対してリアルモードと386 ネイティブモードの割込みハンドラを別々にセットする場合などに重要な機能である。

INT21h
AX=2508h　　　セグメントリニアベースアドレスの入手

入力　　　　BX＝ｾｸﾞﾒﾝﾄ セレクタ

出力　　　　正常終了：
　　　　　　　キャリーフラグ＝クリア
　　　　　　　ECX ＝セグメントのリニアベースアドレス
　　　　　　セグメントセレクタが無効な場合：
　　　　　　　キャリーフラグ＝セット

指定されたセグメントのリニアベースアドレスはGDT またはLDT のディスクリプタエントリから取出され、ECX に格納して返される。

INT21h
AX=2509h　　　リニアアドレスの物理アドレスへの変換

入力　　　　EBX＝変換するリニアアドレス

出力　　　　正常終了：
　　　　　　　キャリーフラグ＝クリア
　　　　　　　ECX ＝物理アドレス
　　　　　　リニアアドレスがページテーブルにマッピングされていない場合：

キャリーフラグ＝セット

このファンクションはリニアアドレスを物理アドレスに変換する。変換されるリニアアドレスはセグメントのリニアベースアドレスを入手するためにシステムファンクション：2508h をコールし、それをセグメントの内の希望するオフセットに加算することによって得られる。

このシステムコールはリアルモードと386 ネイティブモードの混在プログラム中のリアルモードコードのアドレスを得るために使用してはならない。この目的にはファンクション：250Fh を使用する必要がある。

INT21h
AX=250Ah　　セグメントの終端での物理メモリのマッピング

入力　　ES ＝変更するセグメントのLDT 内のセグメントセレクタ
EBX＝マッピングするメモリの物理ベースアドレス。4 KBの倍数でなければならない。
ECX＝マッピングすべき 4 KB物理メモリページの数

出力　　正常終了：
キャリーフラグ＝クリア
EAX ＝マッピングされたメモリのセグメント内のオフセット
エラー終了：
キャリーフラグ＝セット
EAX ＝エラーコード
＝8 ：ページテーブル作成用メモリが不十分
＝9 ：セグメントセレクタが無効

指定されたセグメントのリミットはマッピングされたページ数だけ増やされ、システムページテーブルは指定された物理ページをマッピングするように設定される。メモリは必ずセグメントの終りにマッピングされるため、マッピングされたメモリの返されるオフセットは必ずシステムコールが行われる前のセグメントリミット＋1とおなじである。

このシステムコールは、通常セグメントを設定してメモリマップドハードウェア装置にアクセスするために使用される。通常の方法はゼロページを要求するINT 21h ファンクション48h コールすることにより、別のLDT セグメントディスクリプタを割り当て、次にこのファンクションをコールして希望する物理メモリにマッピングする方法である。このシステムコールは既存のセグメントの終りにメモリをマッピングするのにも使用することができる。

INT21h　ファンクション49h(セグメントの解放）またはINT21 ファンクション4Ah(セグメントのメモリ割当ての変更）のシステムコールはこのファンクションによりマッピングされた物理メモリを持つセグメント上で許されることに注意すること。

割り当てられたメモリかマッピングされたメモリを既に持っているセグメントに対してこのコールを行った場合、LDT セグメントディスクリプタのセグメントリニアベースアドレスが変わることがある。この副作用はアプリケーションプログラムにはみられない。

INT21h
AX=250Ch　　ハードウェア割込みベクタ入手

入力	なし
出力	キャリーフラグ＝クリア AL＝IRQ0－IRQ7ハードウェア割込みのベース割込みベクタ AH＝IRQ8－IRQ15 ハードウェア割込みのベース割込みベクタ

このシステムコールはハードウェア割込み発生時に8259A 割込みコントローラによって生成された割込みベクタを返す。MS-DOS下で実行している場合、IRQ8－IRQ15 のベース割込みベクタは、常に割込みベクタ 40hであり、IRQ0－IRQ7のベース割込みベクタは割込みベクタ 48hである。-HWIVEC コマンドスイッチは割込みの異なるブロックを選択するときに使用する。

INT21h
AX=250Dh　　リアルモードリンク情報の入手

入力	なし
出力	キャリーフラグ＝クリア EAX ＝リアルモードから386 ネイティブモードルーチンをコールするリアルモードDOS-Extenderプロシジャのアドレス EBX ＝モード間コールデータバッファのﾘｱﾙﾓｰﾄﾞアドレス ECX ＝モード間コールデータバッファのサイズ（単位：バイト） ES:EDX＝モード間コールデータバッファの386 ネイティブモードアドレス

このシステムコールの返す情報は、リアルモードと386 ネイティブモードのコードが混在するプログラムによって使用される。EAX とEBX で返されるリアルモードアドレスは上位16ビットのリアルモードセグメントアドレスと下位16ビットのセグメント内オフセットから構成される。リアルモードと386 ネイティブモードのコードの混在については第８章を参照のこと。

モード間コールデータバッファのサイズは０から64KB（デフォルトは０）であり、-CALLBUFコマンドラインスイッチで設定される。このバッファはLOW メモリに割り当てられることが保証され、常にDOS-Extenderで使われることはなく、アプリケーションプログラムで使用される。

アプリケーションプログラムのリアルモードコードは、EAX で返されるDOS-Extenderのリアルモードルーチンを FAR コールすることにより、同一プログラム内の386 ネイティブモードプロシジャを呼び出すことができる。この時スタックは以下のように設定される。

SS:SP --> PWORD　　：386 ネイティブモードプロシジャのFAR アドレス
　　　　　DWORD　　：リアルモードのパラメタブロック（後述）へのFAR ポインタ
　　　　　パラメータ：386 ネイティブモードプロシジャに渡されるパラメータ

そして、DOS-Extenderルーチンの FARコールが行われる。386 ネイティブモードプロシジャが制御を受け取ると、スタックは次のようになる。

SS:ESP --> QWORD　　：FAR プロシジャコールのリターンアドレス
　　　　　パラメータ：386 ネイティブモードプロシジャに渡されるパラメータ

386 ネイティブモードプロシジャの処理が終了すると、FAR リターンで抜ける。

386 ネイティブモードプロシジャが制御を受け取る時、汎用レジスタとプロセッサフラグの内容は、リアルモードでDOS-Extenderルーチンを呼び出した時と同じ値である。セグメントレジスタの内容はDOS-Extenderルーチンに渡されるパラメータブロックに依存する。パラメータブロックポインタのセグメント部分にゼロの値が渡されると、DS,ES,FS,GS レジスタは386 ネイティブモードプログラムが起動する時と同様にオリジナルな値がロードされる。これ以外では、次に説明する形式のパラメータブロックの値がロードされる。

ｵﾌｾｯﾄ	サイズ	意味
00h	WORD	386 ネイティブモードでのDSレジスタの値
02h	WORD	386 ネイティブモードでのESレジスタの値
04h	WORD	386 ネイティブモードでのFSレジスタの値
06h	WORD	386 ネイティブモードでのGSレジスタの値

386 ネイティブモードのスタックはリアルモードのスタックと同一のメモリを示すように設定される。セグメントセレクタ：0060h （MS-DOSが使用する先頭の１MBをマップ）がSSレジスタにロードされる。

リアルモードプロシジャが再び制御を受け取った時、すべての汎用レジスタとプロセッサフラグの内容は、386 ネイティブモードプロシジャがリターン命令を実行した時と同一の値である。スタックはDOS-Extenderルーチンを呼び出した時と同じ内容に設定される。ただし、スタック上のパラメータブロックへのポインタと386 ネイティブモードプロシジャへのポインタはゼロに設定される。オリジナルのリアルモードセグメントレジスタの値が復元される。パラメータブロックがDOS-Extenderルーチンに渡された場合は、386 ネイティブモードプロシジャがリターンする時にセグメントレジスタに入っていたセグメントセレクタは、パラメータブロック内に格納される。

INT21h

AX=250Eh　　リアルモードプロシジャの呼出し

入力	EBX =呼出すリアルモードプロシジャのアドレス ECX =386 ネイティブモードスタックからリアルモードスタックへコピーする4バイトのDWORD の数	
出力	正常終了： キャリーフラグ=クリア セグメントレジスタの内容はすべて変更されない。すべての汎用レジスタは、リアルモードプロシジャによってロードされた値を持つ。 エラー終了： キャリーフラグ=セット EAX =エラーコード =1：リアルモードスタックスペースが不十分	

このシステムコールは386 ネイティブモードからリアルモードプロシジャを呼び出すのに使用される。EBX で渡されるリアルモードプロシジャのアドレスは、上位16ビットにリアルモードのセグメントパラグラフアドレス、下位16ビットにセグメントのオフセットが入る。リアルモードのプロシジャが制御を受け取る時は、すべての汎用レジスタとプロセッサフラグの内容はシステムコールが実行された時と同じ値を持っている。DSセグメントレジスタはCSレジスタと同じ値がロードされ、ES,FS,GSレジスタの値は保証されない。SS:SP はDOS-Extenderが割り当てたスタックバッファ（デフォルトは0バイトで、-ISTKSIZE コマンドラインスイッチで設定可能）を指すように設定される。指定された数のパラメータがリアルモードスタックにコピーされる。リアルモードルーチンが制御を受け取った時、スタックは以下のよう設定さている。

```
SS:SP -->  DWORD    ：リアルモードのFAR リターンアドレス
           パラメータ：リアルモードのスタックにコピーされたパラメータ
```

リアルモードプロシジャが処理を完了すると、FAR リターンを実行してその処理を終了する。386 ネイティブモードルーチンが再び制御を受け取ると、すべての汎用レジスタとプロセッサフラグの値は、リアルモードプロシジャを呼び出した時の値である。ただし、キャリーフラグ（システムコールのリターン情報として利用）は除く。セグメントレジスタは386 ネイティブモードでのオリジナルな値に復元される。

INT21h
AX=250Fh　　386 ネイティブモードアドレスのMS-DOSアドレスへの変換

入力　　ES:EBX＝変換する386 ネイティブモードアドレス
　　　　ECX ＝データ長（単位：バイト）

出力　　正常終了：
　　　　　キャリーフラグ＝クリア
　　　　　ECX ＝MS-DOSアドレス
　　　　エラー終了：
　　　　アドレスがMS-DOSパーティションにマッピングされていない場合：
　　　　　キャリーフラグ＝セット

このファンクションは可能な場合に386 ネイティブモードアドレスをMS-DOSのLOW メモリアドレスに変換する。セグメント：オフセットアドレスは ECXで返される（上位16ビットにセグメントパラグラフアドレス、下位16ビットにオフセットがそれぞれ入る）。

このファンクションは物理メモリアドレスを得るというよりは、リアルモードと 386 ネイティブモードのコードが混在するプログラムのリアルモードのコードで使用するアドレスを求めるために使われる。

また、このシステムコールはMS-DOSのLOW メモリ内で連続で、そしてLOW メモリにすべて納まっているデータバッファやコードブロックの確認にも利用できる。 ECX中の入力された長さ値が０より大きい場合は、このコールは指定された長さ全体は連続でないか、またはLOW メモリにない時にエラーを返す。

INT21h
AX=2510h　　リアルモードプロシジャの呼出し（レジスタ指定）

入力　　EBX ＝呼び出すリアルモードプロシジャのアドレス
　　　　ECX ＝386 ネイティブモードからリアルモードにコピーされるスタックのワード数
　　　　DS:EDX＝パラメータブロックへのポインタ

出力　　正常終了：
　　　　　キャリーフラグ＝クリア
　　　　　すべてのセグメントレジスタは変更されない
　　　　　EDX ＝変更なし
　　　　　他の汎用レジスタはリアルモードプロシジャの設定した値が入る。
　　　　　プロセッサフラグはキャリーを除きリアルモードプロシジャの終了時の値となる。

リアルモードのレジスタの値はパラメータブロックで返される。
エラー終了：
キャリーフラグ＝セット
EAX ＝エラーコード
＝1：リアルモードスタックスペースが不十分

このファンクションは386 ネイティブモードからリアルモードプロシジャを呼び出すシステムコール：250Eh と類似している。異なる点は、セグメントレジスタを含むリアルモードのすべてのレジスタが指定され、そして返されることである。DS:EDXでポイントされるパラメータブロックは以下の形式である。
386 ネイティブモードのコードが混在するプログラムのリアルモードのコードが使用する。

オフセット	サイズ	内　容
00h	WORD	リアルモードのDSの値
02h	WORD	リアルモードのESの値
04h	WORD	リアルモードのFSの値
06h	WORD	リアルモードのGSの値
08h	DWORD	リアルモードのEAX の値
0Ch	DWORD	リアルモードのEBX の値
10h	DWORD	リアルモードのECX の値
14h	DWORD	リアルモードのEDX の値

リアルモードのプロシジャが制御を受け取ると、ファンクション：250Eh と同じようにスタックが設定される。パラメータブロックで指定された値が対応するレジスタにロードされる。その他のレジスタやプロセッサフラグはこのシステムコールが実行された時と同じ値を持つ。

386 ネイティブモードプロシジャがコール後再び制御を受け取ると、オリジナルの386 ネイティブモードセグメントレジスタすべての値が復元され、オリジナルのEDX （パラメータブロックのポインタ）が復元され、キャリーフラグを除くプロセッサフラグとその他の汎用レジスタの内容は、リアルモードプロシジャによって設定された値を持つ。パラメータブロックは、リアルモードプロジシャによってセグメントレジスタとEDX に残されたリアルモードの値を含み、そしてリアルモードのEFLAG のキャリーフラグを含む全体の値はパラメータブロックのEAX の位置に置かれる。

INT21h
AX=2511h　　リアルモード割込みの発行（レジスタ指定）

入力　　DS:EDX＝パラメータブロックへのポインタ

出力　　正常終了：

EDX は変更なし
他の汎用レジスタはリアルモード割込みハンドラが設定した値が入る。
プロセッサフラグはリアルモード割込みハンドラの設定した値が入る。
リアルモードレジスタの値はパラメータブロックで返される。

このシステムコールはセグメントレジスタを含むすべてのレジスタを指定してリアルモード割込みの発行を可能にする。DS:EDXで示すパラメータブロックの形式は以下の通りである。

ｵﾌｾｯﾄ	サイズ	内　容
00h	WORD	割込み番号
02h	WORD	リアルモードのDSの値
04h	WORD	リアルモードのESの値
06h	WORD	リアルモードのFSの値
08h	WORD	リアルモードのGSの値
0Ah	DWORD	リアルモードのEAX の値
0Eh	DWORD	リアルモードのEDX の値

パラメータブロックで指定された割込みはリアルモードで発行される。割込みの発行前に、パラメータブロックで指定されたレジスタの値が対応するレジスタにロードされる。他のレジスタの値は変更されない。
このシステムコールの処理完了後に386 ネイティブモードルーチンが制御を再び受け取ると386 ネイティブモードの元のセグメントレジスタの値とEDX （パラメータブロックポインタ）の値が復元され、他のすべての汎用レジスタとプロセッサフラグがリアルモードの割込みハンドラで設定された値となる。パラメータブロックの値はリアルモード割込みハンドラで設定されたセグメントレジスタやEDX の値が設定される。
このシステムコールは他のDOS-Extenderシステムコールのようにキャリーフラグにリターン状態を返さないことに注意すること。代わりにリアルモードの割込みハンドラが設定したプロセッサフラグの値を返す。

INT21h
AX=2512h　　デバッグのためのプログラムのロード

入力　　DS:EDX＝プログラム名（ゼロで終了する文字）
　　　　ES:EBX＝パラメータブロックのポインタ
　　　　ECX　＝ＬＤＴバッファのバイトサイズ

出力　　正常終了：

キャリーフラグ＝クリア
EAX ＝ＬＤＴ内のセグメントディスクリプタの番号
エラー終了：
キャリーフラグ＝セット
EAX ＝エラーコード
＝２：ファイルがないまたはパスが無効
＝５：アクセスを拒否された
＝８：メモリの不足
＝10：環境が無効
＝11：無効なファイル形式
＝128：ＬＤＴが小さ過ぎる

このシステムコールはデバッガで使われ、別のプログラムをEXECするために使用してはならない。プログラムをメモリ中にロードし、呼出し側のLDT バッファ内のセグメントディスクリプタを初期化する。このLDT はロードしたプログラム用のLDT とするためにデバッガによって設定される。ES:EBXが示すパラメータブロックの形式は以下の通りである。

《入力》

ｵﾌｾｯﾄ	サイズ	内　容
00h	DWORD	環境文字列のオフセット
04h	WORD	環境文字列のセグメント
06h	DWORD	コマンドテール文字列のオフセット
0Ah	WORD	コマンドテール文字列のセグメント
0Ch	DWORD	LDT バッファのオフセット（サイズはECX で指定）
10h	WORD	LDT バッファのセグメント

《出力》

ｵﾌｾｯﾄ	サイズ	内　容
12h	WORD	ロードされたプログラムのPSP のリアルモードパラグラフアドレス
14h	WORD	ﾘｱﾙ/386ﾈｲﾃｨﾌﾞﾓｰﾄﾞﾌﾗｸﾞ(0=ﾘｱﾙﾓｰﾄﾞ、1=386 ﾈｲﾃｨﾌﾞﾓｰﾄﾞ)
16h	DWORD	EIP の初期値
1Ah	WORD	CSの初期値
1Ch	DWORD	ESP の初期値
20h	WORD	SSの初期値
22h	WORD	DSの初期値
24h	WORD	ESの初期値
26h	WORD	FSの初期値
28h	WORD	GSの初期値

環境文字列のセグメントセレクタが０の場合、親の環境を受け継ぐ。０でない場合、それは０で終わる文字列の並びであり、環境文字列の終りはさらに０の文字が付く。コマンドテール文字列は設定する必要があり、最初のバイトは文字列の長さ（それ自身とキャリッジリターンは除く）を指定する。文字列の長さに続く文字列はキャリッジリターンで終了する。

呼出し側のLDT バッファの大きさは少なくともECX で指定されたサイズでなくてはならない。言い換えれば、ロードされるプログラムで必要とするだけの数のセグメントディスクリプタがLDT バッファ内で初期化される。最小限８個のディスクリプタが初期化される（表3-1 参照）。LDT バッファ内の初期化されないディスクリプタは０となる。システムコールの復帰時、パラメータブロックのオフセット12h から上のフィールドはプログラムの初期状態の値が格納される。

プログラムロード後、MS-DOSのカレントPSP は親プログラム（ デバッガ）のPSP の値である。子プログラム内のPSP にあるCTRL/C、重大エラーおよび終了ベクタは変更されないために、それらはMS-DOSをポイントしている。DOS のDTA バッファは子プログラムのPSP 内をポイントするように設定されるが、DOS-ExtenderはDTA バッファを使用する386 ネイティブモードシステムコールではそれ自身の内部バッファを使用するため、386 ネイティブモードモードでは何の効果はない。

このシステムコールはリアルモード/ 386 ネイティブモードプログラム両方のロードに使用でき、さらにDOS-Extenderを組み込んだ386 ネイティブモードプログラムのロードにも使用できる。プログラムの動作違いの混乱を避けるために、DOS-Extenderとデバッガのバージョンを同一にしておく必要がある。

プログラムをロードするのに十分なメモリがないと、このシステムコールはエラーとなる。デバッガはメモリの使用方法を制限する-MAXDATAスイッチでリンクするか、実行時にメモリを解放するかをしなくてはならない。リアルモードプログラムをデバッグする場合-MINREALまたは-MAXREALスイッチでLOW メモリを十分解放する必要がある。

子プログラムはLDT を使い尽くす可能性もあるため、デバッガはGDT をサーチしてフリーのディスクリプタを見つけ、そのコード、データ、PSP 、環境セグメントをGDT に移動し、そのセグメントレジスタすべてを再ロードすることが必要である。一度このステップを実行すれば、デバッガは子プログラム用のLDT バッファをポイントするようにGDT のセグメント:0028hと0030h 用のディスクリプタを変更することができる。デバッガ用のセグメントをGDT に移動するまで割込みをインタセプトしてはならないことに注意すること。

INT21h

AX=2513h　　セグメントディスクリプタの別名の設定

入力	BX＝別名を指定するディスクリプタ（GDT でもLDT でも良い）のセグメントセレクタ CL＝アクセス許可バイト CH＝USE タイプビット(USE16またはUSE32)
出力	正常終了： 　キャリーフラグ＝クリア 　AX＝別名のゼグメントセレクタ エラー終了： 　キャリーフラグ＝セット 　EAX ＝エラーコード 　　＝8：メモリが不足（LDT がすでに最大サイズになっている、またはLDT 作成用のメモリがない） 　　＝9：BXのセグメントセレクタが無効

このシステムコールは指定したセグメントの別名をLDT 内に生成する。生成された別名のセグメントには、システムコールの呼出し側が指定したアクセス許可バイト（ディスクリプタのバイト5）が設定される。アクセス許可バイトはセグメントに許可されるアクセスと同様に、そのセグメントに許可されるセグメントのタイプを指定する。詳細は参考文献1を参照。呼出し側はセグメントのUSE タイプビット（ディスクリプタのバイト6のビット6）を指定できる。タイプはCHで指定され、USE16 では00h を指定し、USE32 では40h を指定する。CHのビット6以外の値は無視される。

DOS-Extenderは別名の付けられたセグメントを管理していないことに注意すること。別名を持つセグメントがメモリに追加、マッピングおよび削除されたりすると、セグメントの別名用のセグメントディスクリプタはその変化を反映するように更新されない。

INT21h

AX=2514h　　セグメント属性の変更

入力	BX＝変更されるディスクリプタ（GDT でもLDT でも良い）のセグメントセレクタ CL＝新しいアクセス許可バイト CH＝新しいUSE タイプビット(USE16またはUSE32)
出力	正常終了： 　キャリーフラグ＝クリア

エラー終了：
　キャリーフラグ＝セット
　EAX ＝エラーコード
　　　＝9 ：BXのセレクタが無効

このシステムコールは指定したセグメントのアクセス許可バイトとUSE タイプビットを変更する。セグメントディスクリプタ変更後、変更したディスクリプタを参照するセグメントレジスタが新しいセグメント属性を認識出来るように、すべてのセグメントレジスタに値を再ロードする。

INT21h
AX=25C0h　　MS-DOSメモリブロックの割当て

入力　　BX＝割り当てるMS-DOSメモリのパラグラフ数

出力　　正常終了：
　キャリーフラグ＝クリア
　AX＝割り当てられたメモリブロックのリアルモードパラグラフアドレス
エラー終了：
　キャリーフラグ＝セット
　AX＝エラーコード
　　＝7 ：MS-DOSメモリ制御ブロックが破壊された
　　＝8 ：メモリが不十分
　BX＝使用可能なメモリブロックの最大サイズ（単位：パラグラフ）

このファンクションはリアルモードでアクセス可能なメモリを割り当てることに使用できる。その最も共通的な利用方法は、リアルモードと386 ネイティブモードの混在プログラム用である。

デフォルトではすべてLOW メモリはDOS-Extenderによってアプリケーション用に使用されるため、このシステムコールはエラーとなる。プログラムリンク時またはDOS-Extenderの実行時に、-MINREALおよび-MAXREALスイッチを指定してLOWEメモリを確保する必要がある。詳細は7.2.1 を参照のこと。

INT21h
AX=25C1h　　MS-DOSメモリブロックの解放

入力	CX＝解放するメモリブロックのリアルモードパラグラフアドレス
出力	正常終了： 　キャリーフラグ＝クリア 　EAX ＝破壊される エラー終了： 　キャリーフラグ＝セット 　AX＝エラーコード 　　＝7 ：MS-DOSメモリ制御ブロックが破壊された 　　＝9 ：CX内のメモリブロックアドレスが無効

ファンクション：25C0h で以前割り当てられたLOW メモリブロックを解放する。

INT21h
AX=25C2h　　MS-DOSメモリブロック割当ての変更

入力	BX＝新たに要求されたブロックサイズ（単位：パラグラフ） CX＝変更するメモリブロックのリアルモードパラグラフアドレス
出力	正常終了： 　キャリーフラグ＝クリア 　EAX ＝破壊される エラー終了： 　キャリーフラグ＝セット 　AX＝エラーコード 　　＝7 ：MS-DOSメモリ制御ブロックが破壊された 　　＝8 ：メモリが不十分 　　＝9 ：CX内のメモリブロックアドレスが無効 　BX＝使用可能なメモリブロックの最大サイズ（単位：パラグラフ）

ファンクション：25C0h で、以前割り当てられたLOW メモリブロックのサイズの変更を行う。

付録F プロセッサ例外

80386 プロセッサは、プログラムエラーを検出すると、プロセッサ例外を発生させる。例外発生時 386 | DOS-Extenderは、例外の種類と発生したプログラムアドレスを示すメッセージを表示して、プログラムを終了させる。終了させられたプログラムは、MINIBUG または 386 | DEBUG デバッガによってデバックしなければならない。これらのデバッガは、例外が発生すると制御権を獲得し、ユーザがメモリやレジスタを調査し、変更できるようにする。

80386 で起こり得るプロセッサ例外を以下にまとめて示す。さらに詳しくは、リファレンス 1 を参照されたい。

例外名	割込み番号	内　　容
Divide Error	0	DIV またはIDIVの命令を使って、0で除算した。
Debug Exception	1	この例外は、デバッガによって使用されるので、アプリケーションレベルでは絶対に発生しない。
One Byte Interrupt	3	この例外は、one-bye INT3命令によって発生し、デバッガが使用する。
Overflow	4	この例外は、INTO命令の実行時とオーバーフローフラグの設定時発生する。
Array Bounds Check	5	この例外は、BOUND 命令の実行時と、オペランドが指定されたリミットを超えた時に発生する。
Invalid Opcode	6	違法な命令コードを実行しようとした。通常、データの真中にジャンプすることによって発生する。

例外名	割込み番号	内　容
Coprocessor Not Available	7	コプロセッサがない場合に80287 または80387 の命令をエミュレートするために使用される。この例外は、コプロセッサ命令が用いられ、CR0 レジスタのEMビットが設定されると発生する。
Double Fault	8	例外の処理中に別の例外が発生した。
Coprocessor Segment Overrun	9	コプロセッサ命令オペランドを80287 または80387 コプロセッサに転送中にページまたはセグメントの違反が発生した。
Invalid TSS	10	タスク切換え中に無効なタスク状態セグメントが検出された。 386｜DOS-Extenderはタスク切換えを使用しない。
Segment Not Present	11	セグメントレジスタにロードされているセグメントセレクタ値によって参照されるLDT またはGDT 内のセグメント記述子が不在と印されている。
Stack Fault	12	SSセグメントレジスタを使用する命令が、スタックセグメントのリミットを超えた。通常、スタックオーバーフローが原因である。
General Protection Fault	13	セグメントリミット違反、誤ったセグメントセレクタ値、セグメントセレクタのプロテクション設定違反を含む種々の条件によって発生することがある。
Page Fault	14	不在マークのついたページをアクセスしようとした、またはページテーブルの欄のプロテクション設定に違反しようとした。 386｜DOS-Extender下では、通常、プログラムセグメント内のヌルポインタによって参照することが原因である。
Coprocessor Error	16	PC/AT またはPS/2との互換機では絶対に発生しない。MS-DOSシステムでは、この割込みはBIOSシステムコールとして使用される。

付録G 参考文献

1. Intel Corporation, 80386 Programmer's Reference Manual, 注文番号230985, 1986年

2. Intel Corporation, 80387 Programmer's Reference Manual, 注文番号231917, 1987年

3. Ray Duncan, Advanced MS-DOS, Microsoft Press, Redmond, Washington, 1986年

4. Chris H. Pappas and William H. Murray III, 80386 Microprocessor Handbook, Osborne McGraw-Hill, Berkeley, California, 1988年

5. Stephen P. Morse, Eric J. Isaacson, and Douglas J. Albert, The 80386/387 Architecture, John Wiley & Sons, Inc., 1987年

6. Weitek Corporation, WTL 1167 Floating Point Coprocessor Preliminary Data, 1987年 8 月

7. 富士通FMR シリーズ 徹底解析マニュアル ＢＮＮ社編

索引

386 | DOS-Extender
ユーザーズマニュアル

英数・記号

あ行

か行

さ行

た行

な行

は行

ま行

ら行

わ行

FM TOWNS

386 | DOS-Extender™
ユーザーズマニュアル

81SP-0072-1-0

発 行 日　1989年10月
発行責任　富士通株式会社

Printed in Japan